Hybrid Statistical, Machine Learning, and Deep Learning Models for Fault Diagnosis and Prognosis in Condition-based Maintenance

Kamyar Azar

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

March 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Kamyar Azar**

Entitled: **Hybrid Statistical, Machine Learning, and Deep Learning Models for Fault Diagnosis and Prognosis in Condition-based Maintenance**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Amin Hammad

_____ External Examiner
Dr. Hossein Hashemi Doulabi

_____ Examiner
Dr. Amin Hammad

_____ Supervisor
Dr. Farnoosh Naderkhani

Approved by _____
Abdessamad Ben Hamza, Chair
Department of Concordia Institute for Information Systems Engineering

_____ 2022

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Hybrid Statistical, Machine Learning, and Deep Learning Models for Fault Diagnosis and Prognosis in Condition-based Maintenance

Kamyar Azar

Maintenance has always been an essential and inseparable part of manufacturing and industrial sectors. Generally speaking, maintenance strategies aim to prevent asset failures/downtimes to protect investments and to provide a safe working environment. With the recent growth in sensor and data acquisition technologies, a rich amount of condition monitoring data has become available in manufacturing and industrial sectors. Consequently, there has been a recent surge of interest in using more advanced solutions, especially those based on Machine Learning (ML), Reinforcement Learning (RL), and Deep Learning (DL) models, to utilize such extensive and high-quality data in the maintenance domain. In this context, the thesis proposed different ML and hybrid models for prognostic and health management purposes to further advance the maintenance field. In particular, we conducted the following three studies: In the first work, a hybrid and semi-supervised framework is designed based on the hazard rate of the system. The proposed framework can extract the hidden state of the system without domain knowledge. To evaluate the efficacy of the proposed method, a real dataset is used where optimal maintenance policies are obtained based on the extracted states via RL. In the second study, a DL-based model is proposed to predict the hazard rate of the underlying system. As opposed to its statistical counterparts, the proposed predictive model does not assume any linear relationship between the sensors' measurements, and is capable of learning from censored data. In the last study, we investigated application of the proposed methods on high-dimensional data such as images. The proposed methods achieved promising results illustrating their great potential to be used in real-world applications.

Acknowledgments

First and foremost, I would like to express my most sincere gratitude and give my warmest thanks to my supervisor, Prof. Farnoosh Nadarkhani, without whom I would not have been able to complete this thesis. Her utmost support, immense knowledge, and insightful guidance from day one made this journey brighter and more enjoyable. I would also like to thank Prof. Amin Hammad and Prof. Hossein Hashemi Doulabi for their precious time and this great opportunity to have them as my committee members.

Besides my supervisor, I would like to thank my family abroad that I miss immensely. Especially the most beautiful girl in the world, my little sister, who stepped into this world and brought hope, color, and joy in my life. I also wanted to thank my friends and all the people who supported and accompanied me in this stage of my life. Without their sincere support, this would not have been possible.

Last but not least, as once a wise man said, ”I wanna thank me.” I wanted to thank myself for enduring all the difficult times, not losing hope, and consistent efforts toward closing this chapter of my life.

Contents

List of Figures viii

List of Tables x

List of Abbreviations xi

1 Thesis Introduction 1

1.1 Contributions . . . 3

1.2 Thesis Organization . . . 6

2 Background & Literature Review 7

2.1 Condition-based Maintenance . . . 7

2.2 Machine Learning and Deep Learning Applications in CBM . . . 9

2.2.1 Overview of Machine Learning and Deep Learning Models . . . 9

2.2.2 Remaining Useful Life Prediction . . . 14

2.2.3 Cox Proportional Hazard Model . . . 16

2.2.4 Concordance Index . . . 18

2.3 Decision-making Models in CBM . . . 19

2.3.1 Statistical based Models . . . 20

2.3.2 Reinforcement Learning based Models . . . 21

3 Semi-Supervised Clustering-based Method for Fault Diagnosis and Prognosis: A Case Study 23

3.1 Problem Statement . . . 24
3.2 Dataset Description and Preparation . . . 25
3.2.1 The Cox's PHM Model . . . 28
3.3 Proposed Hybrid and Semi-Supervised Framework and Experimental Results . . . 30
3.3.1 Label Construction Stage . . . 31
3.3.2 Reinforcement Learning-based Cost Minimization . . . 35
3.3.3 DRL-based Cost Minimization . . . 39
3.3.4 Prediction Model Training . . . 41
3.4 Summary . . . 52

4 HazNet: Non Linear and Deep Hazard Architecture for Maintenance Management 53
4.1 Problem Statement . . . 54
4.2 Dataset Description . . . 56
4.3 Proposed Model . . . 56
4.3.1 Learning from Censored Data . . . 57
4.3.2 Proposed Model Framework . . . 58
4.4 Experimental Results . . . 60
4.5 Summary . . . 64

5 Deep Learning-based Survival Analysis for a System Subject to Stochastic Degradation using High-Dimensional Condition Monitoring Data 66
5.1 Dataset Description . . . 67
5.2 Model Name . . . 67
5.2.1 Dimension Reduction Technique: Convolutional Autoencoder . . . 69
5.3 Results and Conclusion . . . 74
5.4 Summary . . . 80

6 Summary and Future Research Directions 83
6.1 Summary of Thesis Contributions . . . 83
6.2 Future Research . . . 86

Bibliography **87**

List of Figures

Figure 2.1 (a) Folded and unfolded LSTM network. (b) LSTM Cell 12
Figure 2.2 CNN Network 14
Figure 3.1 (a) Hazard rate corresponding to the 11^{th} engine. (b) Baseline hazard (BH) versus Hazard Risk (HR) associated with the 11^{th} engine. 30
Figure 3.2 Individual Clustering. 32
Figure 3.3 Aggregated clustering 33
Figure 3.4 Space-action space 37
Figure 3.5 (a) Steady-state action plot. (b) Steady-sate cost plot. 38
Figure 3.6 The block diagram of the DRL model. 39
Figure 3.7 (a) Variation of cumulative rewards versus different epochs. (b) Steady-sate cost plot. 40
Figure 3.8 SVM decision boundaries with following hyperparameter configurations: (a) 2-degree polynomial kernel, (b) 3-degree polynomial kernel, (c) 4-degree polynomial kernel, (d) 5-degree polynomial kernel, (e) 6-degree polynomial kernel, (f) 7-degree polynomial kernel, (g) 10-degree polynomial kernel, (h) RBF kernel and auto gamma parameter, (i) RBF kernel and scale gamma parameter, (j) Sigmoid kernel and auto gamma parameter, (k) Sigmoid kernel and scale gamma parameter. 43
Figure 3.9 KNN decision boundaries with the following hyperparameter configurations: (a) one neighbor, (b) two neighbor, (c) three neighbor, (d) four neighbor, (e) five neighbor. 44

Figure 3.10 Naïve Bayes with different algorithms: (a) gaussian Naïve Bayes, (b) complement Naïve Bayes. 44
Figure 3.11 Random Forest training history and decision graph. 45
Figure 3.12 Max voting decision graphs considering three scenarios. 46
Figure 3.13 Optimizing the CBM decision based on Stacking. 46
Figure 3.14 Optimizing the CBM decision based on Feed Forward Neural Network. 48
Figure 3.15 LSTM decision graphs. 49
Figure 4.1 The proposed LSTM network architecture. Stacked Dense layers can be used after LSTM cells to make the network deeper. 58
Figure 4.2 The proposed hybrid model architecture combines LSTM and 1D-CNN. After concatenating the output of LSTM and 1D-CNN paths, it will be fed into an FFNN network. 59
Figure 4.3 The summary of the average and standard deviation of the cross-validated results using LSTM, 1D-CNN, and the hybrid model for discrete and continuous event variables 63
Figure 4.4 Predicted hazard risk for different engines 65
Figure 5.1 Proposed Model Architecture. 69
Figure 5.2 Convolutional Autoencoder Architecture. 70
Figure 5.3 Convolutional Autoencoder configurations used in this study. 71
Figure 5.4 Original, reconstructed, and compressed images using the first Convolutional Autoencoder configuration $(4-8-1)$ 72
Figure 5.5 Original, reconstructed, and compressed images using the second Convolutional Autoencoder configuration $(16-32-1)$ 73
Figure 5.6 Discrete versus continuous event detector performance comparison 76
Figure 5.7 Encoder versus No Encoder performance comparison 78
Figure 5.8 n_past$=5$ versus n_past$=10$ performance comparison 79
Figure 5.9 Performance comparison between all models 81
Figure 5.10 Predicted hazard risk for different image streams 82

List of Tables

Table 3.1 Summary of C-MAPSS Dataset Description. 27

Table 3.2 Estimated Coefficients for C-MAPSS Dataset. 30

Table 3.3 Summary Results of the RL approach. 38

Table 3.4 Train accuracy average, standard deviation (SD), and test accuracy. 42

Table 3.5 LSTM Models' Structure. 47

Table 3.6 Evaluation results obtained under different maintenance cost and policy scenarios. . 50

Table 4.1 LSTM Results 61

Table 4.2 1D-CNN Results 62

Table 4.3 Hybrid Model Results 62

Table 4.4 Network structure used for illustration purposes 64

Table 5.1 Dataset Description 67

Table 5.2 LSTM and Dense Blocks configurations 75

List of Abbreviations

CBM	Condition-based Maintenance
CM	Condition Monitoring
BH	Baseline Hazard
HR	Hazard Risk
DL	Deep Learning
ML	Machine Learning
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
DNN	Deep Neural Network
RL	Reinforcement Learning
MDSS	Maintenance Decision Support System
PHM	Proportional Hazard Model
RUL	Remaining Useful Life
PM	Preventive Maintenance

Chapter 1

Thesis Introduction

Under today's highly competitive business environment, it is critical and of paramount importance that manufacturing and industrial sectors operate at their full potential. Recent heightened enthusiasm towards the Industrial Internet of Things (IIoT) and Industrial Artificial Intelligence (IAI) coupled with developments in smart sensor technologies have resulted in simultaneous incorporation of several advanced Condition Monitoring (CM) technologies within manufacturing and industrial sectors. Capitalizing on the aforementioned importance of IIoT and IAI, Condition-based Maintenance (CBM) can be considered as the enabling mechanism for companies to maintain a competitive advantage over their competitors. The CBM is considered the state-of-the-art maintenance policy that uses CM data to track the health state of the underlying system, predict potential failure point(s), and prevent unwanted failure occurrences. Previously, due to the lack of CM data and low computation power, CBM could not be implemented to its full potential. However, recent advancements in learning and sensing technologies, especially in the domain of Machine Learning (ML) and Deep Learning (DL), allow the development of more complex CBM models. This is mainly due to abundantly accessible ML/DL models and their unique power in analyzing and processing High-Dimensional and Multi-model Streaming (HDMS) CM data [Accorsi et al., 2017; Amruthnath and Gupta, 2018; Ayo-Imoru and Cilliers, 2018; Coraddu et al., 2016; Paolanti et al., 2018; Zhao et al., 2016; Zhu et al., 2019].

A great proportion of literature devoted to the development of ML/DL models within the context of CBM focus on Remaining Useful Life (RUL) prediction. Researches developed several

novel models based on well-known ML/DL models such as Artificial Neural Network (ANN) [Lall et al., 2016; Saon et al., 2010], Convolutional Neural Network (CNN) [Babu et al., 2016; Yang et al., 2019], Long Short-Term Memory (LSTM) [Park et al., 2020], Restricted Boltzmann Machine (RBM) [Deutsch and He, 2016], and hybrid models in which they combine different models like Deep Belief Network-Feedforward Neural Network (DBN-FNN) [Deutsch and He, 2017], CNN-LSTM [Ren et al., 2020], and CNN-Bidirectional LSTM (CNN-BiLSTM) [Wang et al., 2020], [Remadna et al., 2020]. Furthermore, a set of ML algorithms including Linear Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), Gradient Boosting method, AdaBoost is used in [Mathew et al., 2017] for RUL prediction. Although the proposed models to predict RUL are promising, RUL has limitations for systems with catastrophic failure consequences. On the other hand, the hazard rate is a reliable and sufficient indicator that can be used for maintenance decision-making. Researchers have been using a traditional and statistical model to compute the hazard rate to develop optimal maintenance policies under different assumptions. However, considering the nature of the maintenance data, interpretation of its features, and limitations of the traditional and statistical models, developing a specific DL-based hazard prediction model in the maintenance domain has been neglected.

It deserves mentioning that the actual condition or hidden state of a system under monitoring is not observable in many cases. Researchers have been using different approaches to define the state of the system [Lee and Pan, 2017; Miao and Makis, 2007; Naderkhani and Makis, 2016; Yang et al., 2017; Zhan et al., 2006]. However, the availability of the rich CM data and the surge of interest in developing unsupervised ML/DL models have proposed a great opportunity to couple ML/DL-based models with the event-triggered CM data to define the state of the system without human intervention or domain knowledge. Moreover, conventional and statistical-based system monitoring techniques cannot cope with the abundance of CM data and high-dimensional data, like images. On the contrary, the emergence of Reinforcement Learning (RL), which is a policy optimization method in the ML domain, provides a great potential to utilize such an efficient method to obtain optimal maintenance policies from extensive CM data. Such framework could also be used to develop an intelligent Maintenance Decision Support System (MDSS) furtherly.

Briefly, the following three researches are the main focus of this thesis:

- Augmenting statistical and ML models to construct the states of the system based on its hazard rate and find the optimal maintenance policy. The main goal is to propose a framework to build a smart MDSS regarding the advent of extensive and high-dimensional data. In this regard, the thesis proposes an MDSS that combines statistical, ML, and RL methods for fault diagnostic and prognostic considering event-triggered CM data.

- The performance of the statistical methods tends to decrease when more data is available. Moreover, these methods assume unrealistic pre-assumptions about the system under study most of the time. These pre-assumptions may prevent these models from being used for real-world problems. Capitalizing on the insufficiency of the RUL to be a basis of maintenance decision-making for some systems and inadequacy of the statistical models, DL models are applied to predict the hazard rate of a system in this thesis.

- Images are among the most frequently available high-dimensional data nowadays due to their rich amount of information and easy-to-use imaging devices in the real world. This implies a great potential in the maintenance domain to use images to infer the system's health. However, directly measured features are not available in images as opposed to the other sensors' measurements like temperature and force that can be used directly in a failure prediction model. Moreover, failure data is hard and expensive to collect in the maintenance domain. Therefore, in this thesis, a DL-based model is proposed that uses images to predict the hazard rate of a system.

1.1 Contributions

Generally speaking, maintenance is an inevitable part of the manufacturing/industrial sectors that ensures the equipment operates under safe conditions and benefits these sectors by reducing the machine downtime, increasing the asset lifetime, and workplace safety. Therefore, the main objective of this thesis is to propose a framework to develop an intelligent hybrid MDSS based on the hazard rate of a system that uses statistical, ML, DL, and RL methods. Three different case studies are conducted to present such a framework. The main contributions of my thesis research

work are briefly outlined below.

(i) Semi-Supervised Clustering-based Method for Fault Diagnosis and Prognosis: A Case Study: In this study, a semi-supervised hybrid MDSS is proposed that combines the statistical and ML methods to extract the hidden states of the system based on the hazard rate of the system and RL methods to achieve the optimal maintenance policy based on the computed states. In brief, contributions of this research work can be summarized as follows:

- Providing a semi-supervised clustering and RL/DRL-based framework for efficient interpretation and proper use of CM data along with the age of the system for fault diagnostic and prognostic.
- Using hazard function to design an alternative maintenance model, which is a more generalizable and sufficient measure compared to the RUL.
- Developing an innovative methodology to estimate the state of the system without domain-specific knowledge or time-consuming and cost-inefficient assumptions.

(ii) HazNet: Non Linear and Deep Hazard Architecture for Maintenance Management: In the second study, a novel DL-based model is proposed to predict the hazard rate of the system using sensors' measurements by considering the non linear relation of coefficients. In summary, the main contributions of this research work are as follows:

- Capitalizing on the observation that the hazard rate is a more applicable measure than RUL and can be used as a more efficient basis for maintenance decision-making, a non-linear and DL-based variant of the Cox's PHM is proposed for the first time within the maintenance domain. More specifically, unlike the traditional Cox's PHM, the proposed model predicts hazard rate considering a non-linear relation between covariates, which is more generalized and better suited for real-world applications.
- CNN and LSTM-based architectures are proposed to take the inspection time into account and learn from the sequential nature of CM time series. Intuitively speaking, the proposed model could be more satisfying for the maintenance given that, within the context of IIoT, the underlying system is being monitored over time via embedded CM sensors.

- In view of the fact that we may not have that much failure or observed event, in a maintenance dataset, a reformulated loss function is designed. Instead of using a discrete value to determine whether an event has happened or not, a continuous variable is proposed that decides how much an observation can contribute to the model training. In this way, our model is able to learn from the censored data as well.

- Although complex DL-based models could have great performances, training such models usually require a vast amount of data. As reformulated loss function allows censored data to be used in the training process, more complex DL-based models could be utilized. A hybrid DL-based model combining LSTM and One-dimensional CNN was also deployed in this paper. High-performance results have been achieved since a hybrid model can take advantage of the power of both models.

(iii) Deep Learning-based Survival Analysis for a System Subject to Stochastic Degradation using High-Dimensional Condition Monitoring Data: In the line of previous research work, the DL-based hazard model is extended for a system subject to stochastic degradation by considering high-dimensional CM data, i.e., images. This research work makes the following contributions:

- A novel framework based on the LSTM model is proposed to predict the hazard rate from image data. Image data contain rich and high-quality data about the system's degradation and is easy to acquire. However, the statistical methods cannot work with such high-dimensional and large data.

- DL models usually rely on using a substantial amount of data for training purposes. Due to the absence of a considerable amount of failure data in the maintenance domain, a method is proposed to participate the censored data in the model training.

- Concerning the fact that the images are high-dimensional data, Convolutional Autoencoder is used to convert the data into a lower-dimensional space. Using a dimension reduction technique reduces the storage space needed to store the data, decreases the computation time, and removes the redundant features if there are any.

1.2 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 provides a detailed literature review on the topic of interest of this thesis and briefly overviews the background required for the following developments in the next chapters.
- Chapter 3 proposes a semi-supervised clustering-based method for fault diagnosis and prognosis and a framework for an intelligent MDSS.
- Chapter 4 presents the proposed single DL-based models and the hybrid model to predict the hazard risk of a system using sensors' measurements.
- Chapter 5 modifies the proposed model in Chapter 4 to be used on image data.
- Chapter 6 concludes the thesis and discusses some of the potential future research works in this line of research.

Chapter 2

Background & Literature Review

In today's competitive business world, every company observes the urgency to work with full potentials to keep up with competitors and stay, strongly, in the business. Given advancements and evolution of the IIoT, and the advent of smart sensors technologies, CM data becomes abundantly accessible. Availability of rich and real-time CM data together with advancements in ML, in particular, Deep Neural Networks (DNNs), have made it possible to intelligently and more efficiently monitor industrial systems and optimally perform maintenance decision making. Rise of IIoT and IAI have encouraged industrial/manufacturing sectors to deploy intelligent and autonomous systems for production. As a result, CM data can be understood and processed better and more efficiently. It is expected that IIoT and IAI soon become integral component of every business seeking to become successful in the market, and keep up or advance against its competitors. Moreover, it is viable to maintain the integrity of complex systems via an efficient and smart prognostic and health management system. A smart prognostic and health management system is responsible to minimize the cost by reducing failures via monitoring the system continuously, detecting any distinct abnormalities or trends in CM data, and suggesting optimal or near-optimal control and maintenance actions.

2.1 Condition-based Maintenance

In today's globalized, interconnected, and competitive market, it is critical and of paramount importance that the Modern Complex Manufacturing and Service (MCMS) systems, such as aerospace,

transportation, and smart power grid, operate at their full potential with highest achievable reliability. Such systems rely heavily on the operation of their constituent technical components that are subject to random failures due to daily usage and degradation. Failures could lead to a variety of severe consequences ranging from destruction of infrastructures to endangering human lives. To avoid such costly failure, tremendous efforts have to be invested in particular to perform proper maintenance actions. Maintenance has always been a crucial part of a manufacturing system, which assures that MCMS system is working as expected. It composes of necessary actions to ensure reliable and cost-effective operation of the system. Recently, with the new technological advancement such as evolution of the IIoT and the advent of smart sensors, which allows collection of real-time data, there has been a growth in utilizing a specific type of maintenance known as CBM. CBM is composed of state-of-the-art maintenance policies that track the health of the system by utilizing the CM data. It can further predict the potential failure and prevent unwanted failure occurrences in the system.

Research within the context of CBM has been growing rapidly in the recent years focusing on both theory and practical applications. With regards to the latter, e.g., application of the CBM in power system is proposed by Reference [Arab et al., 2016]. In particular, the effect of hurricanes and tropical storms on the reliability of the power system is considered by developing an integrated infrastructure hardening and CBM scheduling model for critical components of the power systems. A novel hybrid CBM policy that jointly considers both perfect and non-memoryless imperfect maintenance is proposed in [Huynh, 2019]. Imperfect maintenance is called memoryless if the effect of the maintenance action at a given time is independent of previous maintenance actions. A new decision role is proposed in [Huynh, 2019] for switching between perfect and imperfect maintenance and the problem is formulated in a partially observable Markov decision process (POMDP) framework. By considering the limitation of inspections in each time epoch due to resource constraints, Reference [Álvarez et al., 2019] proposed a novel CBM policy with non-periodic inspections. Unlike many other researchers, the authors considered a variable maintenance threshold instead of a predetermined one that can lead to a less frequent inspections, and as a result, considerable cost savings. As a system consist of different component and by considering the fact the each component can have its own failure behavior, Reference [Sun et al., 2017] proposed a CBM strategy for

a multi-unit system that can be applied at the component level. More references in can be found in [Hu et al., 2020; Si et al., 2018; Verbert et al., 2018].

2.2 Machine Learning and Deep Learning Applications in CBM

ML is a sub-field of Artificial Intelligence (AI) algorithms where computational statistics are used to allow machines and computers learn automatically from data without explicit programming. With the recent growth of the computation power as well as abundance of CM data, it is now possible to utilize such a powerful tool for different problems. Generally speaking, ML models can be classified into supervised and unsupervised learning techniques. Supervised learning models are used whenever a labeled dataset is available, and the goal is to train a model to predict these labels. Based on the label data type, a supervised learning model can fall into two categories: regression or classification models. However, unsupervised learning models deal with unlabeled datasets and aims to discover hidden patterns in data without any human intervention. Clustering, finding association rules, and dimension reduction techniques are the three main tasks that unsupervised learning methods are used for.

2.2.1 Overview of Machine Learning and Deep Learning Models

In this subsection, the brief review of different supervised and unsupervised ML models are provided. In summary, the traditional ML models are reviewd followed by recent DL-based models.

(1) **Traditional Supervised ML Models**: Conventional supervised ML models consist of K-Nearest Neighbors (KNN) [Cover and Hart, 1967], Support Vector Machines (SVM) [Cortes and Vapnik, 1995], and Naïve Bayes [Lindley, 1958; Rish et al., 2001], to name but a few. These models are easy to implement and are more interpretable and less complex than other types of supervised ML models.

(2) **Ensemble Learning Models**: Ensemble learning models are meta-algorithms that incorporate the results of heterogeneous or homogeneous base ML models to make more precise predictions and reduce the variance of the predictions. Such models consist of Max Voting,

Stacking, and Random Forest. With regards to the first type models, these models are more complex and harder to implement, but they can ensemble the result of different models and present more accurate models. A brief description of some of ensemble models are given as follows:

- ***Max Voting***: In this method, multiple models are applied to make predictions for each data point. The predictions by each model are considered as a vote. The final prediction is based on the predictions from majority of the models.

- ***Stacking***: Stacking is another ensemble model that combines different predictions from homogeneous or heterogenous ML models. In this algorithm, there are two levels involved. In the first level, multiple models, known as base-models, are used to make predictions for each row of the data. Then, in the second layer, the previous predictions are used as inputs for a meta-learner model. In other words, a meta-learner uses predictions from base-models as a new feature to combine all of the base-models.

- ***Random Forest***: Random forest is one of the ensemble learning methods that use classification or regression trees as basic learners. This algorithm makes up of many decision trees, and trains each one based on a random sampling of observations during building the trees over a random/limited subset of attributes when splitting the nodes. The final decision of a random forest is made by averaging the output of each tree for a regression problem, and by majority vote for a classification problem.

- K**-means**: K-means [Hartigan and Wong, 1979] is a well-known clustering method that can identify homogeneous subgroups in the dataset. These subgroups are also called clusters. K-means algorithm partition data into clusters such that data points in a cluster are as similar as possible to other data points in the same cluster and as dissimilar as possible to other clusters. It uses similarity measurements like squared Euclidean distance to minimize the within-cluster variances and maximize between-cluster variance.

More recent ML type of models are referred to as DL-based models. The rise of DL models is one of the latest trend in AI because of their powerful potentials to extract useful information

and high-level features from raw dataset. Hence, these models can be a good choice to assess the health of a system. Similar to the ensemble learning models, DL models are more accurate but they are less interpretable and need more time to train. Generally speaking, a DL model consists of blocks or units, called artificial neurons, and the network is trained by minimizing a loss function. In each epoch, a batch of data is fed forward and backward through the network, and at the end of each epoch, the parameters of the network are updated so that the loss function is minimized. This process is called forward and backward propagation, respectively. Two of well-known and widely used DL models for time series prediction are briefly reviewed below.

(1) ***LSTM***: is a special kind of RNNs that is capable of learning from the sequence of data, and are designed to solve the RNNs' vanishing gradient problem [Hochreiter and Schmidhuber, 1997]. In this network, some loops allow information to persist, allowing the model to learn from the sequence of data through time. This loop and an unrolled structure of an LSTM network are shown in Fig. 2.1a. The structure of an LSTM cell is shown in Fig. 2.1b. Each LSTM cell consists of a cell state that is similar to a conveyor belt, where information is added or removed during this process. The flow of the information to the cell state is controlled by three structures called gates. In each LSTM cell, there are the following three gates:

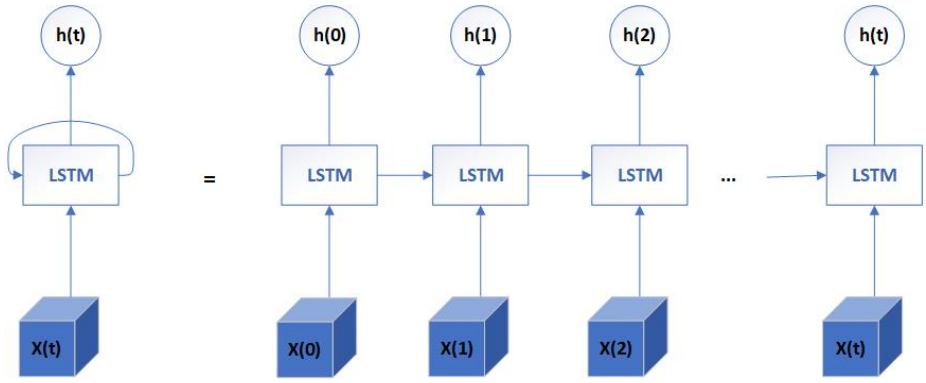
(i) Forget Gate Layer: This gate decides what information should be removed or kept from the cell state.

(ii) Input Gate Layer: This gate decides which values will be updated.

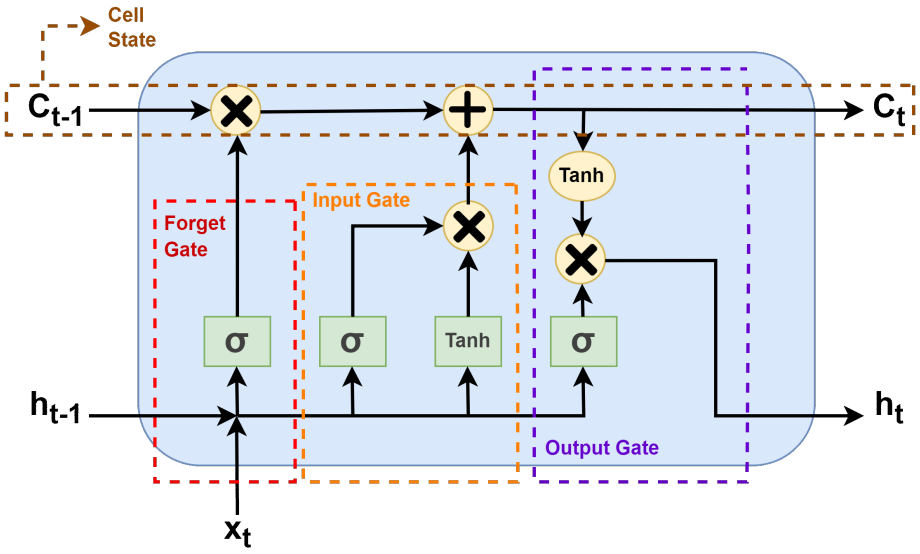
(iii) Output Gate Layer: This gate decides what is the current LSTM cell output. The output of the last LSTM cell, h_{t-1}, and the current input, X_t, flow inside the Forget Gate Layer. This gate looks at h_{t-1} and X_t and using the Sigmoid activation function and

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f), \tag{1}$$

decides what information to keep and what to discarded. In the next step, we decide what new information should be stored in the cell state. This step includes two sub-steps. First, new information is passed to the Input Gate Layer, where for deciding on which new elements to



(a)



(b)

Figure 2.1: (a) Folded and unfolded LSTM network. (b) LSTM Cell

be updated, we use

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i). \tag{2}$$

Secondly, new values are created as follows

$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C). \tag{3}$$

Therefore, in a nutshell, by using a $Tanh(\cdot)$ activation function, new values are created for addition to the cell state, and by using the Input Gate Layer, we decide which values should

be updated in the cell state. Next, the cell state is updated as follows

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{4}$$

Finally, the output will be calculated based on the following two sub-steps: (i) We flow h_{t-1} and x_t inside a sigmoid activation function to decide which parts of the cell state should output as follows

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o). \tag{5}$$

The current cell state is fed into a $Tanh(\cdot)$ activation function to generate the output of the LSTM cell as follows

$$h_t = o_t * \tanh(C_t). \tag{6}$$

(2) ***One-dimensional CNN (1D-CNN)***: is another DL model that originally was proposed to extract information from the image data. However, due to the strength of these models, they can also be used to analyze time series data. Consider that we have a $m * n$ dataset, in which m and n refers to the number of observations and the number of features or variables, respectively. This network will use a window size of length p and width k to choose the input of the network in each run or epoch. In a multivariate time series setting, the window size has the same width as the original dataset, $n = k$. Because of that, we call this network 1D-CNN since the window size cannot move right or left, and it just moves in one direction which is from the first to the last observation. A figure of the network structure is provided in Fig. 2.2 for illustration purposes. other than Input and Output Layer, which most DL networks have, 1D-CNN normally includes Convolutional Layer, Pooling Layer, Flatten Layer, and a Fully Connected Layer.

The first two layers are responsible for extracting and constructing features. The Flatten layer converts the data into a one-dimensional array which is the acceptable shape of data for the next layer. After these three parts, now, we can say that our problem is changed to a simple feed-forward neural network, and we can use a Fully Connected Layer at the end to complete our CNN network. More information about this network and its background can be found

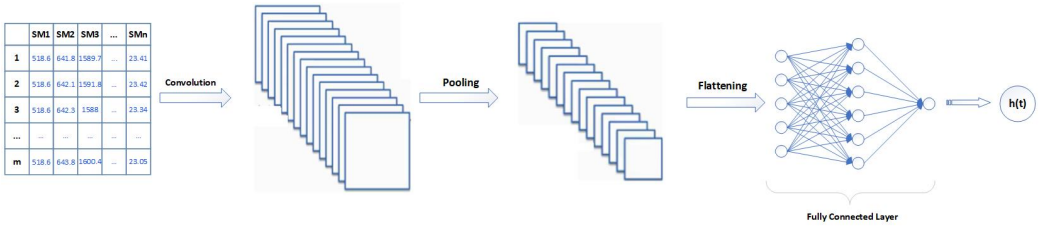


Figure 2.2: CNN Network

in [Wu, 2017].

Recently, there has been a new trend in using ML and DL based solutions for design of CBM strategies because of their unique capabilities in analyzing and processing HDMS CM data [Zhao et al., 2019]. Although using ML/DL techniques in predictive maintenance [Bai et al., 2018; Tian et al., 2011; Wu et al., 2013] is significantly challenging, it has great potentials to introduce exceptional benefits such as superior availability and reliability of assets, capital expenditure decrease (fleet size can be decreased with improved asset availability), and operational expense reduction (e.g., replacement costs). Generally speaking, CBM consists of three main components: (i) Data acquisition; (ii) Data processing, and; (iii) Maintenance decision-making [Jardine et al., 2006]. Although, the first two components have recently been researched extensively, extracted information rarely translated and efficiently/properly used for maintenance decision making.One of the popular ML and DL applications in the CBM context is predicting the length of time a system is likely to operate before it fails. This is also known as RUL prediction. We will briefly review some of the researches in this area alongside their limitations in the following section.

2.2.2 Remaining Useful Life Prediction

ML/DL-based RUL prediction has attracted significant attention recently. Using ML/DL models to predict RUL is one of the recent contributions in this line of research. For instance, a deep bidirectional Long Short-Term Memory (BiLSTM) model is used to estimate RUL for gas turbine engines based on a public dataset provided by NASA [Saxena et al., 2008] in [Wang et al., 2018]. The author shows promising results in comparison with other models like Multi-Layer Perceptron (MLP) and Support Vector Regression (SVR). Another interesting work is presented in [Miao et al., 2019] where the authors proposed a dual-task deep Long Short-Term Memory (LSTM) model for

joint learning of degradation stage assessment and RUL prediction for achieving more robust and accurate result. Authors in [Li et al., 2018] proposed a new DL method based on Convolution Neural Networks (CNN) to estimate the RUL. After comparison with other methods, they show the effectiveness and excellence of their model. Authors in [Yu et al., 2020], developed a data-driven approach based on Similarity-based Interpolation (SBI) to estimate the RUL. In their method, the high-dimensional sensors' measurement is first transformed into a one-dimensional Health Index (HI) that shows the degradation and the health of the system. After that, RUL is estimated using the SBI method. A bidirectional RNN autoencoder-based model is constructed to compute the HI values more robustly in comparison to the RNN autoencoder model that was traditionally used. Furthermore, a domain-specific rule is proposed to more accurately predict the RUL. In addition to proposing a novel one-dimensional HI, Shi and Chehade [Shi and Chehade, 2021], developed a novel Dual-LSTM network that could detect the change point, which is a point in time where the underlying system starts to deteriorate. The authors discussed that detecting the change point is beneficial since it is not reasonable to predict RUL before that, and the data before this change point does not provide potentially useful information. A review of the current literature and status of this line of research, along with the future opportunities can be found in [Xu and Saleh, 2021], and more references can be found in [Rengasamy et al., 2020]-[Wang et al., 2019]. Although the results are promising and highly accurate, it is believed that RUL estimation alone without design of a decision support system is not a sufficient measure for maintenance decision-making. Furthermore, while RUL can provide useful information, it may not be an adequate measure to make decisions solely based on its value. That is because, in the literature, a wide range of papers deal with estimating RUL without considering that in some systems, mostly where a failure can be catastrophic, it is preferable to predict the probability that a machine or a system operates without failure until the next inspection interval. In such systems, relying only on how much time left before observing a failure (i.e., the RUL) is not enough for maintenance operators to decide whether the inspection interval is suitable or not. In this regard, "Hazard Rate" which is a frequently used function to analyze the survival data is preferable. Hazard rate is a sufficient measure and it can address the mentioned problem. Moreover, the RUL can be calculated from the hazard rate, but not vice versa. This function will be discussed more in the next section.

2.2.3 Cox Proportional Hazard Model

Survival function and hazard rate are two frequently used terms to analyze survival data. Survival data analysis is a branch of statistics that analyzes the expected duration of time until one or more events of interest happen. Every survival data is a three-dimensional array that involves the observation time (T), event detector (E), and a set of measurements that are observed during the observation process (X). In the maintenance context, the event of interest is the failure of the underlying system. Therefore, if the failure has occurred, E will be equal to 1, otherwise, E will be equal to 0. In the latter case, it is said that the observation is right-censored, or in the other words, the event would happen later. Dealing with right-censored data is covered well in the literature [Huang and Wellner, 1997; Jia and Jeong, 2020; Koul et al., 1981; Qin and Shen, 2010; Wong et al., 2017].

The survival function $S(t)$ given by

$$S(t) = Pr(T \geq t) = 1 - F(t), \tag{7}$$

indicates the surviving probability of a monitored system beyond t giving that the event of interest (e.g., failure) has not occurred until t. On the other hand, hazard rate, which given by

$$h(t) = \lim_{dt \to 0} \frac{Pr(t \leq T \leq t + dt \mid T \geq t)}{dt}. \tag{8}$$

is the conditional probability that the event of the interest will occur in a very slight time later if the event has not happened until now, or t. It is worth nothing that survival function $S(t)$ can be computed knowing the hazard rate $h(t)$ and vice versa as follows

$$h(t) = \frac{f(t)}{S(t)}, \tag{9}$$

$$S(t) = \exp\left(-\int_0^t h(t)\, dt\right), \tag{10}$$

where $f(t)$ is the Probability Density Function (PDF) of the time to failure. Therefore, either survival function $S(t)$ or the hazard rate $h(t)$ can be used as the basis for monitoring the health state of the underlying system. As discussed in the last section, we will use the hazard rate in this

thesis given its unique characteristics. A Proportional Hazard Model (PHM), referred to as Cox's PHM proposed by [Cox, 1972], is a frequently used model in survival analysis and can be used to formulate and estimate the hazard rate. Such a model considers both the age of the system, or baseline hazard, and the system's degradation, or hazard risk and relates the age of the system and a number of monitored measurements or covariates with the hazard rate. The Cox's PHM model is given by

$$h\left(t, X\right) = h_0(t) \times g\left(X\right), \tag{11}$$

where $h_0(t)$ represents the baseline hazard that considers the age of the system. Term $g(X)$ represents the hazard risk, which takes the system degradation over time into account through the incorporated covariates (X). Commonly, a linear hazard risk is assumed in the literature, i.e.,

$$h\left(t, X\right) = h_0(t) \times \exp\left(\sum_{j=1}^{p} \beta_j X_j\right), \tag{12}$$

where $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is referred to as the coefficients measuring the impact of the covariates on the hazard rate. Linear hazard risk, however, can be a trivial assumption for practical scenarios. Therefore, different variants have been proposed in the literature [Augustin, 2004; Kleinbaum and Klein, 2012; Tian et al., 2005; Wu et al., 2014] to address this issue and taking time into account. Next, we present one of these extensions known as DeepSurv that aims at relaxing a linear assumption for the risk function.

DeepSurv is a nonlinear extension of the Cox's PHM model considering a nonlinear hazard risk. DeepSurv uses a feed-forward neural network to estimate the hazard risk of each input as follow

$$h\left(t, X\right) = h_0(t) \times \left(e^{\hat{g}(X)}\right), \tag{13}$$

where $\hat{g}(Z)$ is the output of the feed-forward neural network. DeepSurv uses the average negative log of the partial likelihood proposed in [Cox, 1975] as the loss function. The idea was first proposed by [Faraggi and Simon, 1995] and it was further improved by [Katzman et al., 2018]. [Katzman et al., 2018] introduced a Cox proportional hazard DNN to model the interactions between a patient's covariates in the healthcare domain. They also provided a personalized treatment

recommendation based on their network results, where a feed-forward neural network is used to estimate the log-risk function. In most of the data on which the original network is trained, only one hidden layer is considered with Scaled Exponential Linear Units (SELU) or Rectified Linear Unit (ReLU) activation function. This could be due to the fact that considerable proportion of the utilized datasets is censored data. By performing several experiments on simulated and real data, they show that DeepSurv outperforms other state-of-the-art survival models in most cases. One of the drawbacks, however, is that the loss function is formulated in such a way that it cannot learn from censored data. In maintenance, since we are observing sensors measurements over time, presence of an abundant number of censored data is inevitable, and there is just a limited number of failure data in most cases. Since training DL models, typically, relies on accessibility of a considerable amount of data, lack of failure data introduces critical challenges for their application within the maintenance domain. Another drawback is that a feed-forward neural network cannot consider the sequential data, therefore, it is not suitable when the observation is time-dependent. To evaluate their model, the authors used a well-known metric used in the survival analysis known as concordance index which will be briefly reviewed in the following subsection.

2.2.4 Concordance Index

Concordance index or C-index, is a commonly used metric to evaluate survival models. It reflects how well a model predicts the order of observing events. In other words, the basic idea here is that if a system failed earlier, it would have a greater risk to fail [Harrell et al., 1982; Uno et al., 2011]. Using a random model results in C-index $= 0.5$, and a perfect model results in C-index $= 1$. The C-index is given by

$$\text{C-index} = \frac{\text{\# of concordant pairs}}{\text{\# of total pairs}}. \tag{14}$$

A pair of observations is said to be concordant if the one that failed earlier, has a greater Risk of Failure (RF). Otherwise, it is said to be discordant. Therefore, when considering right-censored

data, Eq. (14) can be modified as follows

$$\text{C-index} = \frac{\sum_{i,j} I_{T_j<T_i}.I_{RF_j>RF_i}.E_j}{\sum_{i,j} I_{T_j<T_i}.E_j}. \tag{15}$$

Here, for every pair of observations i and j ($i \neq j$), the following three scenarios can occur:

(1) E_i = E_j = 1 and $T_i > T_j$. Meaning that both i and j has failed and i failed later than j. In this case, this pair of observation is concordant if $RF_i < RF_j$ and it is discordant if $RF_i > RF_j$.

(2) E_i = E_j = 0. Meaning that both i and j has censored. In this case, since we do not know which observation has failed earlier, we cannot consider this pair of observation in the C-index computation.

(3) $E_i \neq E_j$. Meaning that one of i or j has been censored and the other one has failed. As an example consider that the former has failed and the latter one has been censored. In this case, if $T_i > T_j$, again we cannot consider this pair of observations in the C-index computation because we do not know which one has failed earlier. On the other hand, if $T_i < T_j$, this pair of observation is concordant if $RF_i > RF_j$ and otherwise, it is discordant.

2.3 Decision-making Models in CBM

In the literature, there is a plenty of existing research on CBM including both statistical and ML-based approaches. Conventional statistical CBM models such as Markov Decision Process (MDP) have been extensively used within the maintenance optimization context to develop optimal maintenance policies. Cox's PHM as described previously is used to formulate and estimate the hazard rate. Recently, due to the lack of statistical methods and the availability of the CM data, researchers have been using the DL-based optimization method, known as RL, to compute the optimal maintenance policy. These two approaches will be reviewed in the next two subsections.

2.3.1 Statistical based Models

In the conventional Cox's PHM, the covariates, that associates the hazard rate with the degradation of the system, are assumed to be consistent over time. Despite that, a time-varying Cox's PHM consists of covariates that can change over time. The parameters of such a model can be estimated with a statistical approach and an optimal maintenance policy can then be achieved. One of the early works in this context is presented by [Makis and Jardine, 1991,9], where the optimal replacement policy is derived in a Semi-Markov Decision Process (SMDP) framework. In this work, the Cox's PHM is considered with a right continuous stochastic covariate process under periodic maintenance. Finally, to minimize the system's long-run expected average cost, a recursive algorithm was developed. In [Wu and Ryan, 2010], the previous discrete-time approximation assumption was relaxed, and optimal policy for a deteriorating system using PHM was proposed. An autoregressive model with the time effect is considered for the first time in [Tang et al., 2015] to formulate the system's degradation. Next, the Mean Residual Life (MRL) is derived based on this autoregressive model. Finally, they determined an optimal maintenance policy under different inspection strategies to minimize the long-run cost of the system by formulating the decision problem in the SMDP framework. The authors in [Duan et al., 2020] designed a two-level cost-optimal Bayesian control chart based on the Hidden Markov Model (HMM) for early failure detection of mechanical equipment. In their work, dependent failure modes are considered, which are referred to as degradation and catastrophic failures. The difference between these two failure modes is that catastrophic failure can happen any time when the system is under operation, but the degradation failure can happen only after the sojourn time in the unhealthy state of the system. In this regard, to model this failure dependency, it is assumed that in the healthy state, the joint distribution of the time to the catastrophic failure and sojourn time follows Marshall-Olking bivariate exponential distribution. The optimization problem for the developed control chart is solved in the SMDP framework, and MRL is calculated using the Bayesian approach. To evaluate their model, they used real multivariate dataset from a milling machine and compared their model with a single sampling interval and a single control limit. In [Zheng et al., 2020], a CBM policy with a dynamic threshold (instead of a constant one) is proposed for a

system subject to minor or catastrophic failures. They assumed multiple actions, namely no maintenance, imperfect maintenance, and preventive replacement, in each inspection epoch and developed a modified policy-iteration algorithm within the SMDP framework to solve their optimization problem. Finally, a complete literature review of CBM models with emphasis on mathematical modeling and optimization approaches is provided in [Alaswad and Xiang, 2017]. An optimal maintenance policy for defense application is presented in [Wong et al., 2010], which is another application of CBM methodology within the prognostic and health management framework. More references can be found in [Jardine et al., 1998]-[Jafari et al., 2017].

2.3.2 Reinforcement Learning based Models

Conventionally, policies and decision support systems developed based on Cox's PHM are statistical in nature. However, given availability of extensive CM and HDMS sensory data, statistical approaches are proficient for most of today's problems. That is because statistical approaches can be used on problems with a few input data and features. However, when the size of input data and the number of features increases, it is hard for such approaches to handle the relationships and correlations inherit in the data. Specifically, statistical solutions for Cox's PHM models pose serious analytical challenges to deal the dimensionality and variety of the data. Moreover, most of the statistical approaches are time-consuming as the number of inputs increase. Finally, most statistical approaches, typically, rely on some unrealistic pre-assumptions rendering their application for real-world problems impractical. As stated previously, the complexity of modern manufacturing systems and exponential growth of CM data in different modalities, make such conventional models impractical. This has resulted in emergence of promising research ideas in the design and application of state-of-the-art AI/ML algorithms that contribute to advancement of maintenance decision making. One of the most widely used ML-based solutions for CBM optimization is known as RL in which a learner, or an agent, learns what actions to take in each state of the system from its interactions with the environment. As a recent example, we can refer to Reference [Yao et al., 2020] where the authors proposed an RL-based solution to find the optimal policy for long-term pavement maintenance planning. The authors discussed that since the state-action space is large, using conventional methods such as dynamic programming is time-consuming and inefficient. Therefore, first, they

have applied Principle Component Analysis (PCA) to reduce the state-space dimension, and then, an Artificial Neural Network (ANN) is developed to approximate the Q-value function. After tuning hyper-parameter values, the optimal maintenance policy is obtained including different types of treatments for a 15-year maintenance planning horizon. Similarly, the authors in Reference [Wei et al., 2020] applied a Deep RL (DRL) model to obtain the optimal maintenance policy with the application to bridge maintenance, where the number of components is large. Due to the infinite nature of the state-action space, the problem cannot be solved by dynamic programming. Therefore, the authors used a Convolutional Neural Network (CNN), as a non-linear approximator, to learn the state-action Q-value. They discussed that the network could learn using either a large historical data source or simulation data. Moreover, the transition probabilities are obtained via both simulations and based on historical data. In Reference [Rocchetta et al., 2019], authors proposed an RL-based and non-tabular solution for the problem of optimal Operation and Maintenance (O&M) scheduling of power grids. They combined Q-learning with an ensemble of ANN to be applicable for large systems with high dimensional state-action spaces. For more information in this line of research, please refer to [Huang et al., 2020]-[Yang et al., 2021]. In this context, since the state-action space is exponentially growing, either dimension reduction techniques (such as PCA) are utilized or DRL for CBM will be used, which do not need dimension reduction. In particular, DRL incorporates DL to efficiently solve high-dimensional MDPs by representing the policy or other learned functions as a neural network.

This completes the review of the existing work in the context of CBM, some of the ML and DL applications in this context, and traditional and state-of-the-art methods to optimize maintenance policies. Furthermore, an introduction to some of the terminologies and concepts used across this thesis is also provided. In the next remaining parts of thesis, the detailed proposed models are provided.

Chapter 3

Semi-Supervised Clustering-based Method for Fault Diagnosis and Prognosis: A Case Study

In this chapter, we propose a semi-supervised clustering-based framework for maintenance decision making based on Cox's PHM model. More specifically, we propose an efficient and novel hybrid semi-supervised model for fault diagnostic and prognostics considering CM data, which combines both statistical approaches and ML algorithms. In the proposed model, we first compute two different components of a time-dependent Cox's PHM model, namely, Baseline Hazard (BH) function, which considers the age of the system, and Hazard Risk (HR) function, which takes into the account the CM data. Then, due to unavailability of labeled data, a semi-supervised method is applied to estimate the state of the system based on its age and sensors' measurements. Consequently, all CM data are used to build the clusters, after which the state-space with its associated transition probability are defined based on the aggregated clustering. The results obtained from the introduced aggregated clustering method are then fed to the RL/DRL module to close the loop and minimize the total cost. The proposed framework provides superior performance from application point of view without relying on pre-knowledge and pre-assumption about the data. Furthermore,

the proposed framework has reduced computational complexity and reduced sensitivity to dimensionality, variety and modality of the data. The proposed model can be further considered as a basis to develop a smart MDSS that can monitor the system via sensors, analyze the state of the system, and make optimal maintenance decisions without any human intervention. The rest of the chapter is organized as follows: Section 3.1 deals with the problem statement. Section 3.2 describes the dataset used in development of the proposed decision support system. Section 3.3 presents the proposed hybrid and semi-supervised ML-based MDSS framework, and provides the implementation study and the results. . Finally, Section 3.4 concludes the chapter.

3.1 Problem Statement

As we discussed earlier, CBM is a state-of-the-art maintenance policy that uses CM data to track the health condition of the system, predict the equipment failure, and prevent the occurrence of costly failures. We consider a machine or a system, which is subject to stochastic degradation and random failures that can happen at any time epoch. The system's health condition is monitored via sensory devices deployed in the system, which provide a large amount of sensor measurements and signals over time. The collected CM data should be processed and analyzed for proper maintenance decision making. CM date can have different formats such as value type, wave form type (such as vibration signals) and high-dimensional types (e.g., image sequences). In this case chapter, we propose a model based on ML-based solutions for diagnostic and prognostic of a system subject to failure, which takes raw CM data to make maintenance decisions based on the state of the system. We consider that the system can be in three distinguishable states, namely, healthy, warning, and failure state. Upon system reaching each state, an appropriate maintenance action should be taken. If the system found to be in the healthy state, no action is required. On the other hand, if the system found to be in the warning state, preventive maintenance or minor repair will be performed. Finally, if the system enters the failure state, corrective maintenance will be carried out.

This completes the problem statement. Next, we present the dataset used to develop and evaluate the proposed MDSS framework. It is worth reiterating, as discussed in Section 2.2.2, that recently the main focus of ML/DL modeling based on the utilized dataset was on RUL prediction. It is,

however, strongly believed that RUL alone without design of a proper decision support system is not sufficient for maintenance decision-making. This chapter aims to address this gap.

3.2 Dataset Description and Preparation

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset is one of the most widely used datasets in the prognostic and health monitoring context. C-MAPSS is a tool available in Matlab for the simulation of commercial turbofan engines under different operational conditions. These operational conditions include altitude, Mach number, and sea-level temperature. Various main elements of an engine such as High-Pressure Compressor (HPC), Low-Pressure Compressor (LPC), High-Pressure Turbine (HPT), Low-Pressure Turbine (LPT), etc., are considered. The package outputs 58 different sensors' measurements that only 21 is considered in the simulation conducted by [Saxena et al., 2008] that this chapter will be based on. These sensors' measurements are as follow:

(1) Total temperature at fan outlet

(2) Total temperature at LPC outlet

(3) Total temperature at HPC outlet

(4) Total temperature at LPT outlet

(5) Pressure at fan inlet

(6) Total pressure in bypass-duct

(7) Total pressure at HPC outlet

(8) Physical fan speed

(9) Physical core speed

(10) Engine pressure ratio

(11) Static pressure at HPC outlet

(12) Ratio of fuel flow to Ps30

(13) Corrected fan speed

(14) Corrected core speed

(15) Bypass ratio

(16) Burner fuel-air ratio

(17) Bleed Enthalpy

(18) Demanded fan speed

(19) Demanded corrected fan speed

(20) HPT coolant bleed

(21) LPT coolant bleed

This dataset includes four subsets of data known as FD001, FD002, FD003, and FD004 and each subset. In this case study, we have used the FD001, FD002,FD003, and FD004 datasets such that each data set split into train and test dataset. In total, 21 number of sensors are installed to each component of the system in order to collect CM data. In total, there are 100 engines under operations, which are subject to random failures. The data for each subset of the C-MAPSS dataset simulates several scenarios of the degradation behavior under three operational conditions. Table 3.1 shows descriptive analysis of the measurements obtained from the 21 sensors.

Pre-processing of data consists of two main steps: (i) Sensor selection, and; (ii) Normalization which are described below:

Sensor Selection: As it can be observed from Table 3.1, some of the sensors' measurements have constant values over time or low standard deviation. Therefore, these sensors are excluded form the model and only 14 sensors were selected, namely, SM2, SM3, SM4, SM7, SM8, SM9, SM11, SM12, SM13, SM14, SM15, SM17, SM20, and SM21.

Normalization: To achieve consistent results, typically, ML/DL models rely on a normalization mechanism before training when features have different ranges. Since the value range of different

Table 3.1: Summary of C-MAPSS Dataset Description.

Sensor ID	Mean	STD	Min	Max
SM1	518.67	0.00	518.67	518.67
SM2	642.68	0.50	641.21	644.53
SM3	1590.52	6.13	1571.04	1616.91
SM4	1408.93	9.00	1382.25	1441.49
SM5	14.62	0.00	14.62	14.62
SM6	21.61	0.00	21.60	21.61
SM7	553.37	0.89	549.85	556.06
SM8	2388.10	0.07	2387.90	2388.56
SM9	9065.24	22.08	9021.73	9244.59
SM10	1.30	0.00	1.30	1.30
SM11	47.54	0.27	46.85	48.53
SM12	521.41	0.74	518.69	523.38
SM13	2388.10	0.07	2387.88	2388.56
SM14	8143.75	19.08	8099.94	8293.72
SM15	8.44	0.04	8.32	8.58
SM16	0.03	0.00	0.03	0.03
SM17	393.21	1.55	388.00	400.00
SM18	2388.00	0.00	2388.00	2388.00
SM19	100.00	0.00	100.00	100.00
SM20	38.82	0.18	38.14	39.43
SM21	23.29	0.11	22.89	23.62

sensors' measurements are different, greater values might influence the model substantially. This may lead to a biased final result. To prevent this problem, it is recommended [Zhang et al., 2016] to scale each feature into the same range of values, which in turn boosts the training speed. However, interpretation of results by using such normalization techniques might become harder, because the scale of the values are changed. Given the aforementioned remarkable advantages that come with normalization, normalize data is used for training purposes. Normalization is performed based on the following model:

$$X_i' := \frac{X_i - \min\{X_i\}}{\max\{X_i\} - \min\{X_i\}}, \tag{16}$$

where X and X' represent the i^{th} sensor measurement before and after normalization. This completes our discussion on dataset description and pre-processing steps. Next, Cox's PHM model is applied on the dataset.

3.2.1 The Cox's PHM Model

As stated previously, the hazard rate can be considered as a sufficient measure for maintenance decision making. Cox's PHM model is a popular model to estimate the hazard rate as follows:

$$h\big(t, Z(t)\big) = h_0(t) \times g\big(Z(t)\big), \tag{17}$$

where $h_0(t)$ represents the Baseline Hazard (BH) that shows the effect of the age on the system's failure, and function $g(\cdot)$ shows the Hazard Risk (HR) that represents the effect of covariates. The BH and HR can be modeled by a Weibull and an exponential distribution as follow:

$$h\big(t, Z(t)\big) = \frac{\beta}{\eta}(\frac{t}{\eta})^{\beta-1} \times \exp\Big(\sum_i \lambda_i \times z_i(t)\Big), \tag{18}$$

where β and η are shape and scale parameters of the fitted Weibull distribution, $z_i(t)$ represents the value of the i^{th} covariate at time t, and λ is the corresponding coefficient.

The first two parameters, namely scale and shape parameters, can be estimated by fitting a Weibull distribution on the failure data. The estimated parameters are as follows:

$$\hat{\beta} = 4.409, \quad \text{and} \quad \hat{\eta} = 225.026. \tag{19}$$

As β is greater than one, the hazard function is increasing, which indicates degradation of the system over time. For systems subject to wear out, Preventive Maintenance (PM) should be taken in order to bring the system to healthy condition and minimize the risk of failure. In addition, if the cost of failure replacement is greater than the cost of PM action, which is almost the case in practical scenarios, using a predictive maintenance policy is justifiable. By using a predictive maintenance policy, the expected number of failures will reduce. Consequently, the expected cost of the failure replacement decreases, which results in the increase of the reliability and availability of the system over time.

We would like to clarify that although the value of β is high, performing preventive maintenance could still be an effective maintenance strategy. According to [Jardine and Tsang, 2005], it is

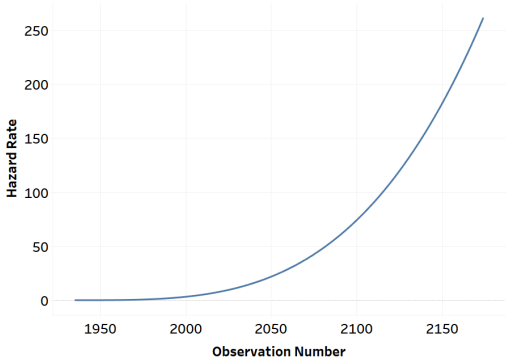
important to note that preventive replacement actions, i.e., actions taken before equipment reaches a failed state, require two necessary conditions: (i) The total cost of the failure replacement must be greater than that of the preventive replacement, and; (ii) The hazard rate of the equipment must be increasing. In the CBM literature, there are several research works considering PM actions while the value of β is high. For example, in [Banjevic et al., 2001], and [Jardine et al., 1987], the estimated β is 4.9 and $[4.03 - 8.97]$, respectively. Authors in [Ma et al., 2016] used a method based on Weibull distribution and deep belief networks to assess the bearing's performance degradation. The estimated shape parameter in this work is considered as 5.067. Moreover, authors in [Zhou and Yin, 2019] developed an opportunistic CBM strategy for offshore wind farms with four different components. Two out of four components have the estimated shape parameters equal to 3. Similarly, in [Lu et al., 2018] another method to obtain the CBM for offshore wind turbines is proposed and three out of four component in the system have the estimated values equal to 3. Finally, authors in [Zheng and Makis, 2020] proposed a CBM policy for a system subject to soft and hard failures. The hard failure hazard rate is described by a PHM model with Weibull-distributed baseline hazard function. The estimated shape parameter is equal 4.63. The results obtained based on the proposed RL approach also confirms the above discussion.

The coefficients of the model are estimated using statistical approaches within the context of time-varying Cox's PHM framework using Maximum Likelihood Estimation (MLE). We would like to clarify that the parameter estimation is implemented based on the "lifelines", which is a complete survival analysis library written in Python [Davidson-Pilon, 2021]. In particular, for parameter estimation, we used the "CoxTimeVaryingFitter" function that fits Cox's time-varying PHM. The utilized function uses Breslow estimation [Xia et al., 2018] considering both effects of age and covariates. It is also worth mentioning that supervised learning models are developed by extracting features to predict a certain label. In other words, the process of extracting and choosing the right set of features (the so called "Feature Engineering") is the most critical component in this context. In our work, choosing Baseline Hazard and Hazard Risk are part of our Feature Engineering scheme. More specifically, the Baseline Hazard and Hazard Risk are utilized as our features. The results are shown in Table 3.2. Finally, for illustration purposes, the hazard rate as well as the BH versus the HR associated with the 11^{th} engine are shown in Fig. 3.1. As it can be seen, the hazard rate

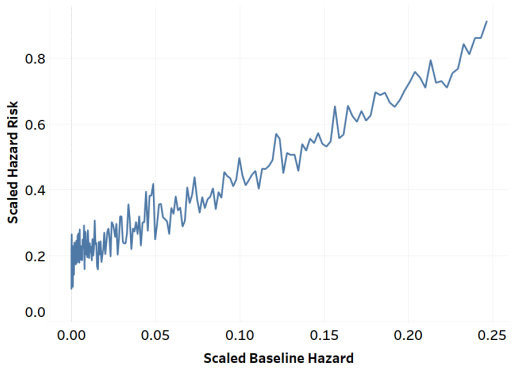
is increasing over time. The next step is to develop a maintenance model based on the calculated hazard rate using ML-based algorithms.

Table 3.2: Estimated Coefficients for C-MAPSS Dataset.

Sensor ID	**Estimated Value λ_i for**
SM2	1.424
SM3	0.021
SM4	0.145
SM7	-0.898
SM8	-1.698
SM9	-0.010
SM11	4.497
SM12	-1.647
SM13	6.201
SM14	0.026
SM15	3.394
SM17	0.245
SM20	-4.873
SM21	-3.720



(a) Hazard rate of the 11th engine



(b) Baseline hazard versus hazard risk of the 11th engine

Figure 3.1: (a) Hazard rate corresponding to the 11th engine. (b) Baseline hazard (BH) versus Hazard Risk (HR) associated with the 11th engine.

3.3 Proposed Hybrid and Semi-Supervised Framework and Experimental Results

In this section, we present the proposed hybrid and semi-supervised ML-based MDSS framework. First, in Subsection 3.3.1, we explain how to construct the label from extracted features,

i.e., BH and HR, based on a semi-supervised learning method with an embedded K-means component. Second, after constructing the labels, we train supervised models to classify the points into the defined categories in the previous step. We trained different types of classification models in this case study, but they fall into three main categories: (i) Traditional Models, (ii) Ensemble Learning Models, and (iii) DL Models. This will thoroughly be discussed in Subsection 3.3.4.

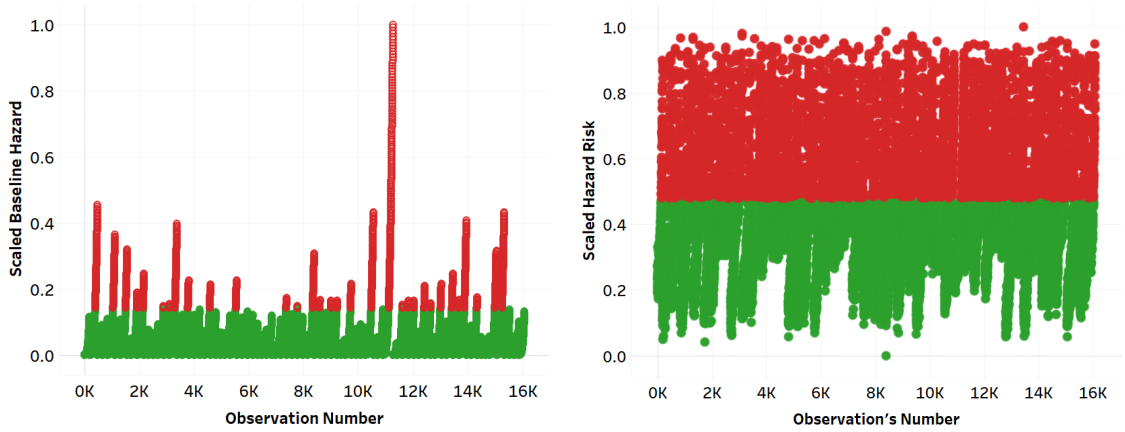
3.3.1 Label Construction Stage

Classification models are supervised learning models, therefore, need outputs or labels. However, sensors' measurements in the C-MAPSS dataset do not have labels and we need to construct the labels (i.e., healthy, warning or failure) autonomously. In this section, we follow our intuitively pleasing procedure to extract and build such data from existing sensors' measurements. To do so, we can take advantage of clustering models. Clustering is an unsupervised learning model that cluster data into different groups. Points in each group are as much similar as possible, and as much different as possible with other groups. This can be achieved by minimizing the inertia or Within Cluster Sum of Square (WCSS) metric. There are several clustering algorithms in the literature. One of the popular and common algorithms in this context is K-mean clustering, which partitions data into the desired number of groups. In this chapter, we utilized a three-step semi-supervised learning procedure and exploit K-means clustering in different steps to cluster data into three groups, namely healthy, warning, and most degraded or failure states. The aforementioned steps are discussed below:

(1) *Step 1. Independent clustering of BH and HR*: In this step, we have used K-means clustering to independently group BH and HR data into two different clusters, namely healthy or unhealthy:

$$C_{\text{BH}} = \begin{cases} 1 & \text{BH is healthy} \\ 0 & \text{BH is unhealthy} \end{cases}$$

$$C_{\text{HR}} = \begin{cases} 1 & \text{HR is healthy} \\ 0 & \text{HR is unhealthy} \end{cases}$$



(a) BH clustering.

(b) HR clustering

Figure 3.2: Individual Clustering.

where C_{BH} and C_{HR} show the clustering label of BH and HR, respectively. The final independent clustering of BH and HR are shown in Figs. 3.2a and 3.2b.

(2) *Step 2. Aggregate Clustering*: In the Step 1, we partitioned BH and HR data into two different clusters independent of each other. In this step, we aggregate the clusters into one single cluster based on below methodology:

(a) If both BH and HR were clustered into the same group, i.e., both were clustered into the healthy or the unhealthy state, the final cluster will be healthy or most degraded/failure state, respectively.

(b) If both BH and HR were clustered into different groups, i.e., BH was clustered into the healthy state and HR was clustered into the unhealthy state and vice versa, the final cluster will be the warning state. In summary, the combination of all possible scenarios can be written as:

$$C_{\text{Total}} = \begin{cases} 2 & C_{\text{BH}} = C_{\text{HR}} = 1 \\ 1 & C_{\text{BH}} \neq C_{\text{HR}} \\ 0 & C_{\text{BH}} = C_{\text{HR}} = 0 \end{cases}$$

where C_{Total} shows the final or aggregated cluster label.

(3) *Step 3. Failure Clustering*: Since the failure time of engines is in a wide range (between

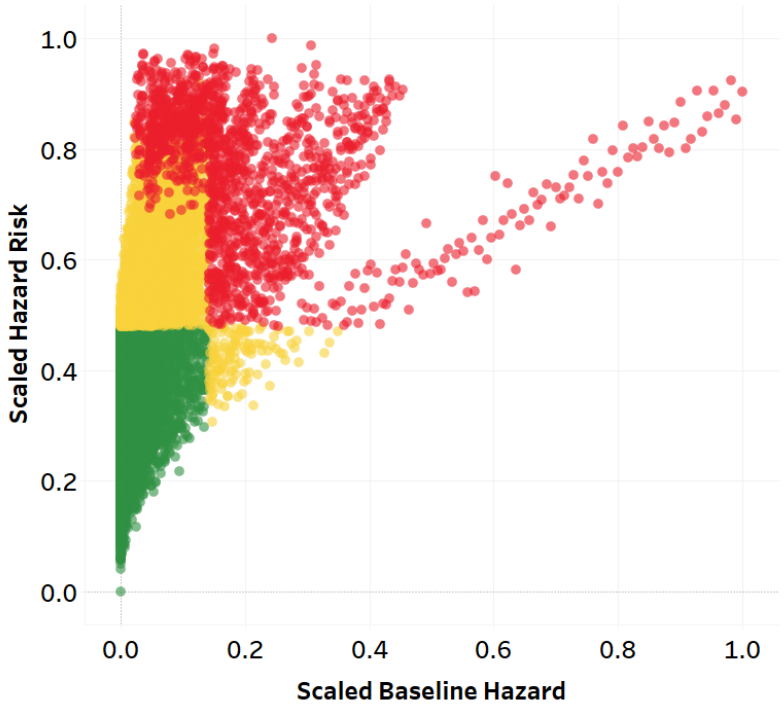


Figure 3.3: Aggregated clustering

128 to 362), it is possible that for some engines that failed earlier, there would not be any most degrade or failure state. If we train the classification model without considering this problem, some potential earlier failures would not be detected. This problem can be handled by manually changing the last d labels of each engine into the most degraded or failure state, i.e.,

$$C_{\text{Total}} := \begin{cases} 0 & \text{The } d \text{ last run-to-failure data} \\ C_{\text{Total}} & \text{otherwise} \end{cases}$$

Finally, we can consider each cluster number as a label for the next step. The final result is shown in Fig. 3.3. This completes our discussion on the labeling stage.

As a final note to our discussion, we would like to clarify the rationale behind independent consideration of HR and BH. Generally speaking, the Cox proportional hazard regression model can be written as $h\big(t, Z(t)\big) = h_0(t) \times g(Z(t))$, where $g(Z(t)) = e^{\beta Z}$. The failure rate is considered to be the product of a baseline failure rate dependent only on the age of the unit and a positive

function $g(Z(t))$ dependent on the values of the stochastic process Z. Thus, the predictors have a multiplicative or proportional effect on the predicted hazard [Makis and Jardine, 1992]. It was shown that the optimal decision is to replace the item whenever the estimated hazard, calculated upon completion of the CM inspection, exceeds a an optimal threshold. Following the above-mentioned mathematical points, Reference [Makis and Jardine, 1992] constructed the following decision strategy. Whenever an inspection is made, the values of the key risk factors are obtained (covariates). These measurements are multiplied by their weighting factors, and then combined to form the composite covariate value, which is marked on the Y-axis. The X-axis shows the age of the item at the time of inspection.

We have followed a similar intuition and constructed our decision graph based on BH and HR such that the BH is showing the working age of an engine on the X-axis and the effect of covariates are plotted across the Y-axis. The idea behind separating BH and HR is coming from the concept of "Proportional" hazards model, where a partial likelihood for β is proposed without involving the baseline function. Suppose we observe $(X_i; \delta_i; Z_i)$ for Engine i, where X_i is a possibly censored failure time random variable; δ_i is the failure/censoring indicator (1 = fail, and 0 = censor), and; Z_i represents a set of covariates. Let $R(t) = \{X_i \geq t\}$ denote the set of engines, which are at risk of failure at time t, called the risk set. Under the proportional hazards assumption, $h(t, Z_t) = h_0(t)e^{\beta Z}$, therefore, the partial likelihood is computed as follows

$$L(\beta) = \Pi_{j=1}^{k} \frac{e^{\beta} Z_j}{\sum_{l \in R(X_j)} e^{\beta Z_l}}. \tag{20}$$

As it can be observed from Eq. (20), the partial likelihood is independent of baseline and it uses the ranks of the failure times. As a final note, it is worth mentioning that in most maintenance decision-making problems, the true state of the system is hidden, however, the observations, i.e., CM data carry partial information about the system's state. In other words, by incorporating CM (sensory) data, the true state of the system becomes partially observable at each sampling epoch. Incorporation of the hazard risk, which itself depends on the instantaneous measurement values (hence conveying partial information about the true state of the system at the execution time), results in a non-static model. We came up with the idea of using an unsupervised learning method to discover the hidden

pattern in the unlabeled data (which otherwise would have been left unused) and categorize the data into different groups. Therefore, by assuming three states in the data (healthy, warning, and failure states), it is logical to assume that their baseline hazard and hazard risk may have similar or close values for observations in a same state or cluster. The parameter estimation is a part of the feature engineering stage of our proposed framework. Every classical ML model needs some features, and feature engineering is a critical component of ML models. Even at times, input to DL models are engineered features.

3.3.2 Reinforcement Learning-based Cost Minimization

We have implemented a novel RL approach based on the results obtained form the clustering method to close the loop and design a MDSS with focus on cost minimization. More specifically, the Q-learning model is developed as the optimization tool for maintenance decision making. Q-learning is a model-free and value-based RL algorithm to learn the value of $Q(s_t, a_t)$ for an action a_t in a given state s_t. The value of $Q(s_t, a_t)$ in each time slot is updated as follows:

$$Q(s_t, a_t) \leftarrow (1-\lambda)Q(s_t, a_t) + \lambda\big(r_t + \gamma \max Q(s_{t+1}, a_{t+1})\big), \tag{21}$$

where γ and λ represent the discount rate and the learning rate, respectively. Term $r_t = \mathcal{R}(s_t, a_t)$ denotes the reward value associated with action a_t at state s_t. In this context, the developed RL model consists of following main components:

(i) **Agent:** *The management system, as the intelligent component of the engine*, acts as the agent and interacts with the environment based on a set of maintenance decisions.

(ii) **State-Space:** The state-space is associated with the *aggregated cluster label*, which consists of healthy ($s_t = 0$), warning ($s_t = 1$), and failure ($s_t = 2$) states.

(iii) **Action-Space:** Given the current state, the optimal action is taken by the agent. The action-space refers to the maintenance operation, discretized into three levels, i.e., no repair ($a_t = 0$), minor repair ($a_t = 1$), and major repair ($a_t = 2$).

(iv) **Reward:** Reward is defined as the maintenance cost. The main goal of the optimal policy

within the RL model is to select the optimal maintenance actions minimizing the total cost or equivalently maximizing the total rewards for a long period of time. In this context, the reward function associated with each action is defined as follows:

$$\mathcal{R}(s_t, a_t) = \begin{cases} 0, & a_t = 0 \\ -20, & a_t = 1 \\ -100, & a_t = 2. \end{cases} . \tag{22}$$

Due to the finite number of state and action spaces, we use Q-learning approach to update the state-action value function in each state, where the size of the Q-table is (3×3). Capitalizing on the importance of the exploration-exploitation trade-off, the ϵ-greedy action selection policy is considered in this context, where ϵ_t in episode t is calculated as follows:

$$\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \exp(\epsilon_d t), \tag{23}$$

where ϵ_{min}, ϵ_{max}, and ϵ_d represent the minimum exploration rate, the maximum exploration rate, and the exploration decay rate, respectively. In this work, it is assumed that $\epsilon_{min} = 1/N_{total}$, $\epsilon_{max} = 1$, and $\epsilon_d = 0.1$, respectively. Term N_{total} is the total number of epochs. In the simulation study, $N_{total} = 200$ number of epochs are considered, where each epoch includes 400 iterations. Finally, the learning rate λ and discount rate γ are considered as 0.01 and 0.99, respectively. The proposed RL model determines the state-action space without incorporation of a data reduction technique such as PCA. More specifically, we have used the results obtained from the introduced aggregated clustering method and fed them to the RL model. In particular, all CM data is used to build the clusters, after which state-space with its associated transition probability are defined based on the aggregated clustering, which are finally fed to the RL model to minimize the total cost.

Fig. 3.4 shows the state-action space for the system under study, including three states 0, 1, and 2, and three actions 0, 1, and 2. In each time, the engine can transit from state i, for $(i \in \{0, 1, 2\})$ to state j, for $(j \in \{0, 1, 2\})$ with action a, $(a \in \{0, 1, 2\})$ with the probability $P^a_{i,j}$. As it can be seen from Fig. 3.4, blue arrows show the transition from state i to state j, when "no repair" action is selected. Similarly, green and red arrows represent "minor repair" and "major repair" actions,

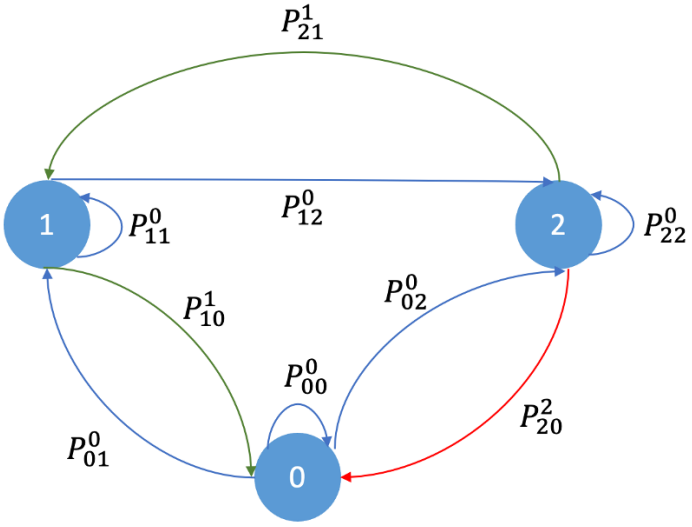


Figure 3.4: Space-action space

respectively. For instance, P_{12}^{0} represents the probability that an engine transits from state 1 to state 2 when no repair action ($a = 0$) is selected. The green arrows show the transition between two sates when minor repair action is performed. We consider the realistic scenario that the effect of minor action is imperfect such that it returns the system to its previous state not the healthy state. Finally, when the system enters the absorbing failure state (state 2), major maintenance action will be performed, which brings the system to healthy state, denoted by red arrow in Fig. 3.4. The transition probability from state i to state j in a single unit time is calculated based on the clustering results as follows:

$$P_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{3} n_{i,k}}, \tag{24}$$

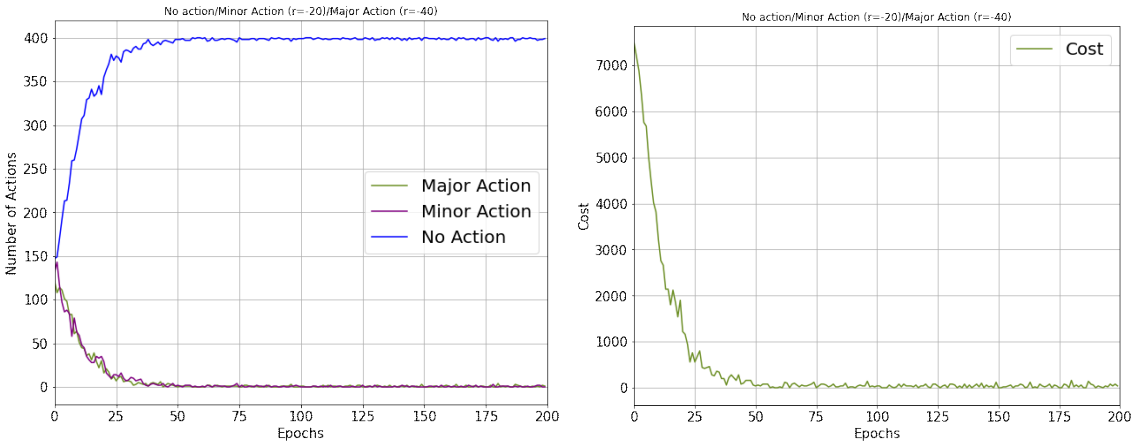
where $n_{i,j}$ is equal to the total number of observations in which the current state is equal to i, and the next state is j. Based on Eq. (24), the transition probability matrix for the system under study is

obtained as follow:

$$\begin{bmatrix} 0.94533 & 0.05452 & 0.00015 \\ 0.13224 & 0.82936 & 0.03840 \\ 0.04433 & 0.04159 & 0.91408 \end{bmatrix} \tag{25}$$

Table 3.3: Summary Results of the RL approach.

Total Cost	**Total Actions**	**Average of Minor Actions**	**Average of Major Actions**
40.75	398.7125	0.5375	0.75



(a) (b)

Figure 3.5: (a) Steady-state action plot. (b) Steady-sate cost plot.

By considering the above-mentioned transition probability formulation, and reward of -10 and 40 associated with the cost of minor and major repairs, respectively, the results of the RL implementation are shown in Table 3.3. The second column of Table 3.3 represents the total actions taken by the agent in each epoch. The last two columns are the average number of minor and major actions taken by the agent when the system reaches the steady state. According to Fig. 3.5, the steady state happens after the 75^{th} epoch where the cost converges. Therefore, we calculated the average of the number of minor and major actions between 75 and 200 epochs.

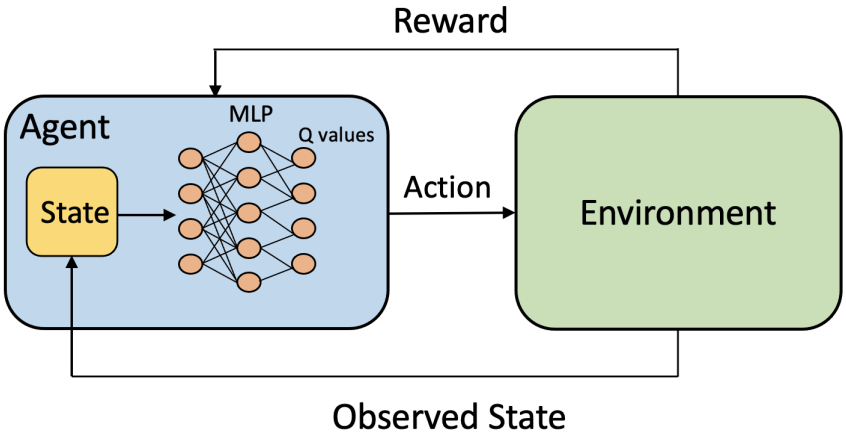


Figure 3.6: The block diagram of the DRL model.

3.3.3 DRL-based Cost Minimization

Here, we consider that the state-space consists of the hazard risk (i.e., a continuous state-space) and its corresponding age of the engine, where the total number of engines is equal to 100. In such a scenario where the state values are continuous, it is not possible to directly use a Q-table for updating the Q values. Consequently, DL models are applied to approximate the state-action value function. As shown in Fig. 3.6, we apply a Multi-Layer Perceptron (MLP) on the Q-Network to estimate Q-values, where the action-space and the reward function are exactly the same as the previous case study. After selecting a maintenance action with the highest value of $Q(s_t, a_t)$, the information associated with the corresponding action is stored in the experience replay, denoted by $D = \{\mathbf{e}_1, \ldots, \mathbf{e}_t\}$, where $\mathbf{e}_t = (\boldsymbol{\phi}_t, a_t, r_t, \boldsymbol{\phi}_{t+1})$. Term $\boldsymbol{\phi}_t = [s_{t-\beta}, a_{t-\beta}, \ldots, a_{t-1}, s_t]$ is the observed state sequence, including state-action pairs between time slots $t - \beta$ and t. The state sequence in replay buffer $\mathbf{e}_m$ is randomly selected to update the weight parameter ξ_t of the learning model based on the Stochastic Gradient Descent (SGD) method. Given ξ_t, the loss function $\mathbb{L}(\xi_t)$ which is the mean-squared error of the optimal and the selected Q-values, should be minimized, where $\mathbb{L}(\xi_t)$ is given by

$$\mathbb{L}(\xi_t) = \mathbb{E}_{\boldsymbol{\phi}_t, a_t, r_t, \boldsymbol{\phi}_{t+1}} \left[\left(Q_T - Q(\boldsymbol{\phi}_t, a_t | \xi_{t-1}) \right)^2 \right], \tag{26}$$

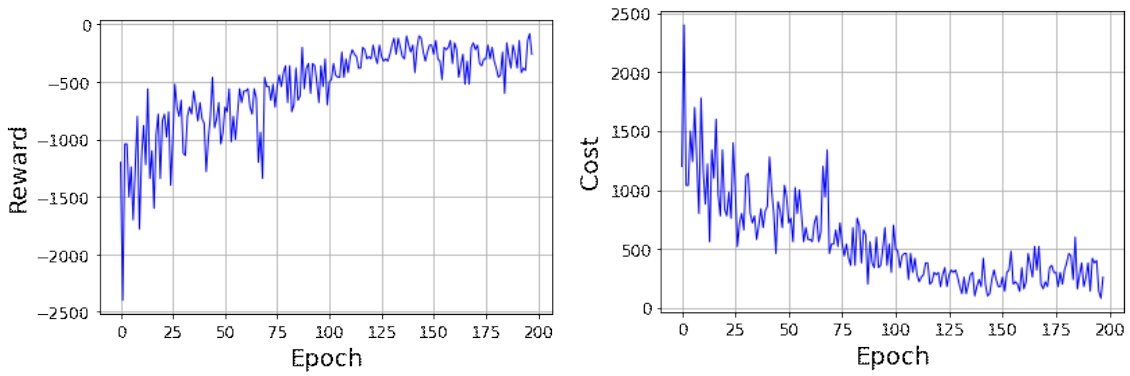
where Q_T as the target optimal Q-value is expressed as

$$Q_T = r_t + \gamma \max_{a_t^{'}} Q(\phi_{t+1}, a_t^{'}|\xi_{t-1})). \tag{27}$$

Given the state s_t, the best action a_t^* is selected based on the ϵ-greedy algorithm with the probability of $(1-\epsilon)$ as follows

$$a_t^* = \arg\max_{a_t^{'}} Q(\phi_t, a_t^{'}). \tag{28}$$

Finally, the reward r_t associated with the corresponding action is calculated and the new experience $\{\phi_t, a_t, r_t, \phi_{t+1}\}$ is stored in the replay memory.



(a) (b)

Figure 3.7: (a) Variation of cumulative rewards versus different epochs. (b) Steady-sate cost plot.

The DRL-based maintenance architecture utilizes a MLP model with one hidden layer with 512 nodes and sigmoid as the activation function. The size of the input and the output layers depend on the state and the action spaces, which is 2 and 3, respectively, where the activation function of the output layer is softmax. The DRL-based maintenance model is performed in 200 epochs. Considering that major maintenance brings the system to the healthy state, we assume that each epoch is terminated if a major repair $a = 2$ is selected by the agent. Fig. 3.7 illustrates the cumulative reward and the maintenance cost of the DRL-based maintenance architecture, respectively. As it can be seen from Fig. 3.7a, increasing the number of epochs increases the cumulative reward, which shows that the model is well trained. As shown in Fig. 3.7b, although the maintenance cost converges using the DRL model, it has the following drawbacks: (i) The DRL model has higher time complexity in

comparison to the RL-based maintenance model. The main reason is that the DRL architecture uses a neural network to approximate the Q-value from the given action and state instead of building a Q-value table. Considering the fact that the continuous hazard risk is used as the state in this model, it takes more time to learn the best action for a given state; (ii) Based on our simulation results, the convergence of the DRL model depends on the definition of the minor repair. For instance, if the minor repair is defined so that the next state is $s_{t+1} = s_{t-5}$, the cost of DRL architecture could not converge to the steady-state, and; (iii) Although considering more layers in the DRL design potentially improves performance of the model, the maintenance cost cannot converge in an acceptable time.

3.3.4 Prediction Model Training

The main objective is to design and develop a hybrid MDSS framework for fault diagnostic/prognostic. The proposed MDSS in place should be able to construct optimal or near-optimal control and maintenance policies for decision-making based on CM data. Hence, we need to monitor the hazard rate of the system continuously to detect any abnormality in data that may indicate a change in the process or the system. As mentioned earlier, PM action is justifiable for a system with an increasing hazard rate. Performing proper PM actions at the right time will increase the system's availability, reliability, and efficiency.

In order to design such a smart system, we need to train a classification model since the outputs are categorical. In our case, a multi-class classification model should be used because we have three different classes, namely healthy, warning, and most degraded or failure class. Furthermore, after constructing the prediction model, each model has a decision graph in which we are interested. To prevent data leakage, first, data should be split into training and test sets. Therefore, in each run, we choose 80% of engines and considered their data as the training set, and the remaining ones as the test set. Then clustering should be fitted on the training set based on the procedure explained in Section 3.3.1. After that, we can transform training and test sets based on the cluster centers found in the previous step. After clustering and constructing the labels, we can train different classification models on the labels. We divided the classification algorithm into three groups: traditional models, ensemble learning models, and DL models.. The models and their results will be discussed more in

depth further.

Besides, the same model can produce different results in each run due to the uncertainty involved in the process. Consequently, we exerted 10-fold Cross-Validation to get relevant results, the measure of bias and variance of the predictions, and to be able to compare different models. The results are shown in Table 3.4. In the following, we will go more into detail and discuss the results.

Table 3.4: Train accuracy average, standard deviation (SD), and test accuracy.

Model	**Average of train accuracy**	**SD of train accuracy**	**Test accuracy**
Random forest with 15 trees in the forest	0.9856	0.0044	0.9826
Stacking classifier	0.9843	0.0049	0.9805
KNN with 3 neighbors	0.9819	0.0049	0.9782
KNN with 1 neighbors	0.9793	0.0052	0.9771
KNN with 5 neighbors	0.9825	0.0036	0.9768
KNN with 4 neighbors	0.9812	0.0034	0.9752
KNN with 2 neighbors	0.9764	0.0043	0.97
SVM with 7-degree polynomial kernel	0.9695	0.018	0.9606
Two-hidden-layer ANN with 25 batch size and 100 epochs	0.9727	0.0102	0.9583
SVM with Radial Bases Function (RBF) kernel and scale coefficient	0.9728	0.0077	0.9569
SVM with 10-degree polynomial kernel	0.9703	0.0161	0.956
Max voting, Version 14	0.9724	0.0073	0.9551
Max voting, Version 19	0.9716	0.0069	0.9541
SVM with 6-degree polynomial kernel	0.9693	0.0049	0.9537
Max voting, Version 10	0.9732	0.0066	0.9528
SVM with 5-degree polynomial kernel	0.9682	0.0048	0.9505
SVM with 4-degree polynomial kernel	0.9689	0.0053	0.9479
SVM with 3-degree polynomial kernel	0.9684	0.0086	0.9434
SVM with 2-degree polynomial kernel	0.9607	0.0098	0.9349
Logistic Regression	0.9507	0.0089	0.9305
SVM with RBF kernel and auto coefficient	0.9525	0.0101	0.9291
SVM with sigmoid kernel and auto coefficient	0.9492	0.0091	0.9278
Gaussian Naive Bayes	0.9014	0.016	0.8847
Complement Naive Bayes	0.6483	0.0342	0.6654
SVM with sigmoid kernel and scale coefficient	0.4013	0.0728	0.3974

Results of Traditional Models

As mentioned previously, these models consist of some well-known and frequently used algorithms in the ML area, including KNN and SVM. Both models have different hyperparameters. To get the best results and compare their decision graphs, we trained each model with different hyperparameters. Figs. 3.8, 3.9 and 3.10 show the decision graph of KNN, SVM, and Naïve Bayes, respectively, with different hyperparameters' configuration. As it is clear, KNN with 3 neighbors, SVM with the 7-degree polynomial kernel, and Gaussian Naïve Bayes have the best results in each category.

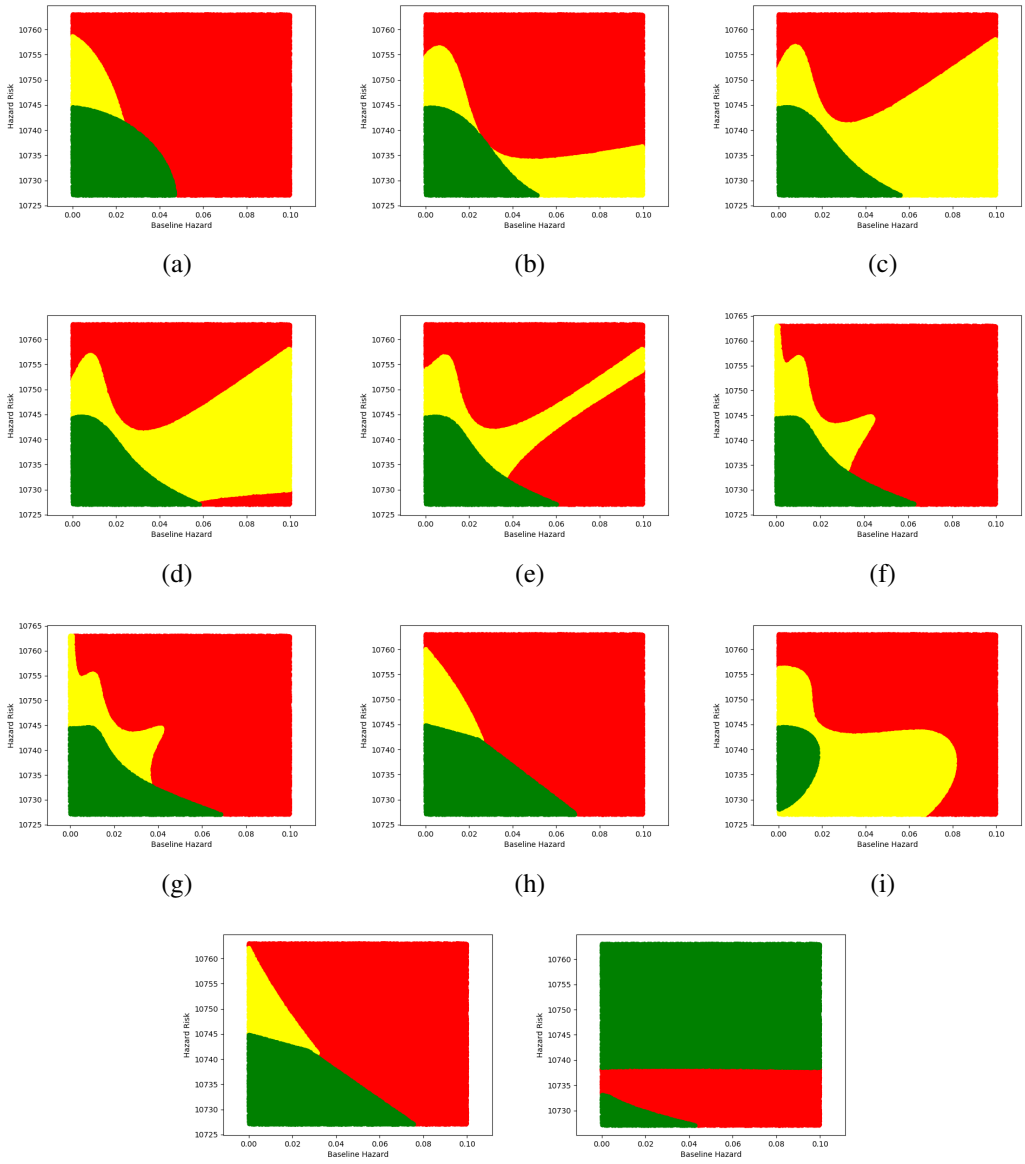
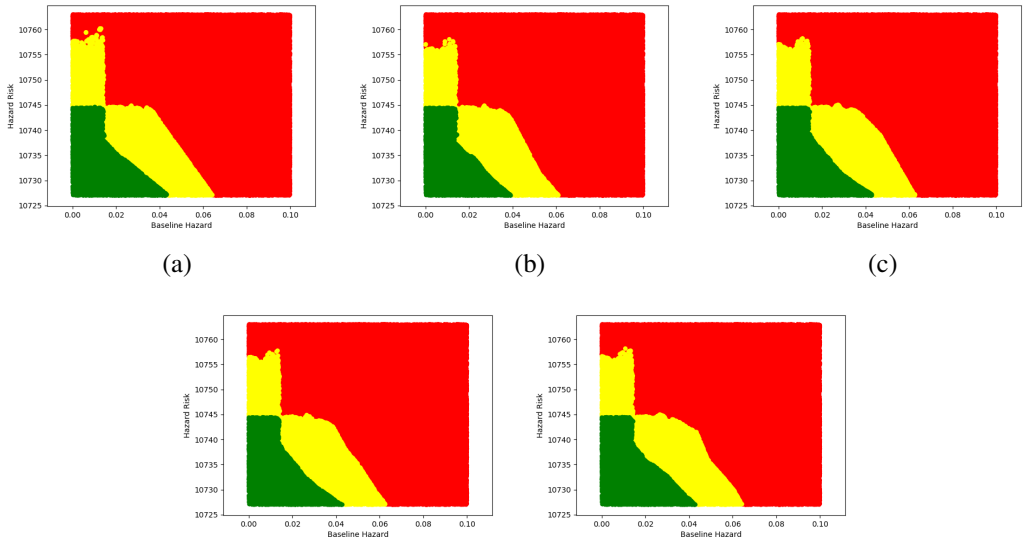


Figure 3.8: SVM decision boundaries with following hyperparameter configurations: (a) 2-degree polynomial kernel, (b) 3-degree polynomial kernel, (c) 4-degree polynomial kernel, (d) 5-degree polynomial kernel, (e) 6-degree polynomial kernel, (f) 7-degree polynomial kernel, (g) 10-degree polynomial kernel, (h) RBF kernel and auto gamma parameter, (i) RBF kernel and scale gamma parameter, (j) Sigmoid kernel and auto gamma parameter, (k) Sigmoid kernel and scale gamma parameter.

Results of Ensemble Learning Models

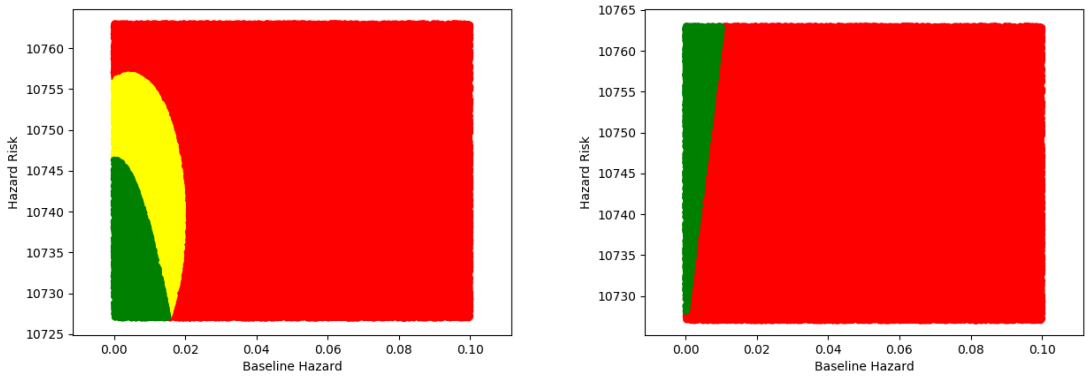
Ensemble learning is a hybrid learning system. As mentioned in the previous section, the basic idea behind ensemble learning is to train different models, known as base models or weak learners,



(a) (b) (c)

(d) (e)

Figure 3.9: KNN decision boundaries with the following hyperparameter configurations: (a) one neighbor, (b) two neighbor, (c) three neighbor, (d) four neighbor, (e) five neighbor.

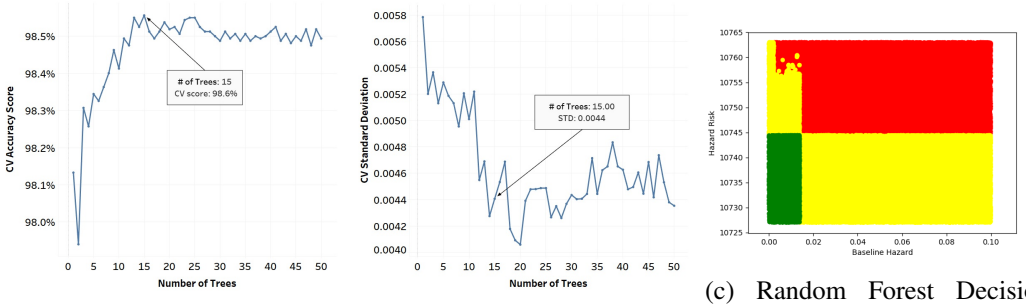


(a) (b)

Figure 3.10: Naïve Bayes with different algorithms: (a) gaussian Naïve Bayes, (b) complement Naïve Bayes.

and aggregate their predictions into one single output. By applying ensemble models, it is expected to enforce the prediction precision and obtain more accurate results. In this study, we have developed three ensemble models including Random Forest, Max Voting, and Stacking which the results are presented as follows:

Random Forest Results: We trained Random Forest with a number of trees in the range of [1, 50]. Figs. 3.11a and 3.11b show 10-fold cross-validated accuracy score and standard deviation for the different number of trees in the forest, respectively. The best results are obtained by using 15 trees. The final Decision graph is shown in Fig. 3.11c.



(a) Random Forest CV score (b) Random Forest CV std (c) Random Forest Decision Graph

Figure 3.11: Random Forest training history and decision graph.

Max Voting Results: Here, different classification models are trained on the same training data, and then, these models are used to make predictions for each data point. Each prediction will be considered as a vote, and finally, the data point will be classified into a class that has the majority of the votes. In this study, we considered three different scenarios:

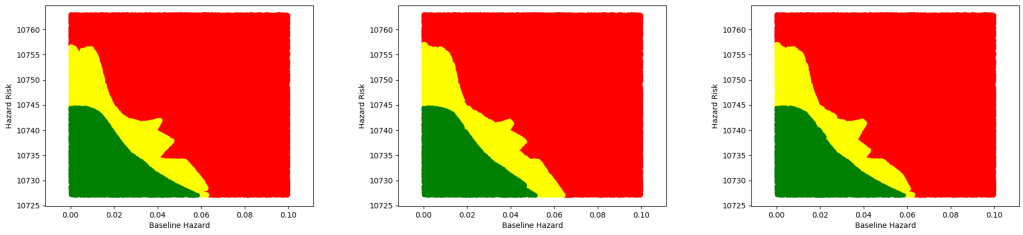
(1) **Scenario 1**: In this scenario, we have considered all the traditional models in Table 3.4 and Random Forest.

(2) **Scenario 2**: In this scenario, we have considered Random Forest, KNN with 1, 3, and 5 neighbors, Logistic Regression, SVM with 2, 3, 4, 6, 7 degrees polynomial kernel, and SVM with RBF kernel with scale and auto coefficients, SVM with sigmoid kernel and auto coefficient, and Gaussian Naive Bayes.

(3) **Scenario 3**: In this scenario, we have considered Random Forest, KNN with 5 neighbors, Logistic Regression, SVM with 2, 3, 4, 5, 6, 7, 10 degrees polynomial kernel.

The decision graph of each scenario is shown in Fig. 3.12.

Stacking Results: As mentioned in Subsection 2.2.1, stacking for classification problems, like Max Voting, considers heterogeneous base models. Each model is used to predict each data point,



(a) Scenario 3 (b) Scenario 2 (c) Scenario 1

Figure 3.12: Max voting decision graphs considering three scenarios.

and these predictions are used as inputs for a meta-model or meta learner. In this study, we considered KNN with 3 neighbors, SVM with the 7-degree polynomial kernel, and Random Forest with 15 trees as base models (best models based on obtained results) and Logistic Regression as the meta-model. The decision graph is shown in Fig. 3.13.

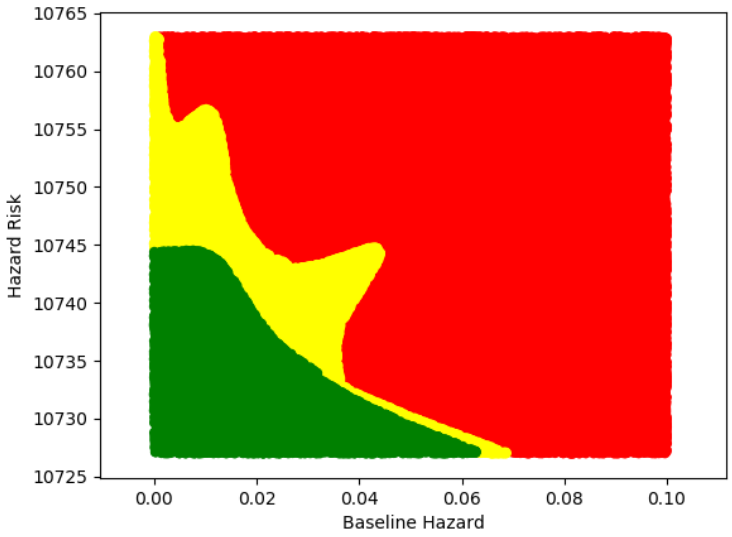


Figure 3.13: Optimizing the CBM decision based on Stacking.

Results of DL Models

Two DL models, namely, ANN and LSTM, are developed. ANN model with 25 batch size and 100 epochs is shown in Fig. 3.14. The configuration of this network is composed of two hidden layers with 6 nodes in each layer, and hyperbolic tangent as the activation function of such layers. LSTM is a refined model of ANN, which can be used to analyze sequence data, like time series.

Since run-to-failure data can be considered as a time series, it is beneficial to use LSTM in our case. We trained three LSTM models with different structure as follows:

Table 3.5: LSTM Models' Structure.

	Number of Hidden Layers	**Number of Nodes in each Hidden Layer**	**Hidden Layer's Activation Function**
First Structure	5	64	relu
Second Structure	5	64	tanh
Third Structure	5	128	relu
Forth Structure	5	128	tanh
Fifth Structure	3	128	relu
Sixth Structure	3	128	tanh
Seventh Structure	3	64	relu
Eighth Structure	3	64	tanh

We cannot plot LSTM's decision graphs like other plots since such models are dependent on time and previous sensor measurements value. Therefore, we content ourselves with plotting the performance of LSTM models in the existing dataset. The results are shown in Fig. 3.15. It is worth mentioning that all these models are developed based on FD001 data set. Similar analysis have been performed considering the other three data sets, i.e., FD002, FD003, and FD004, which are omitted here to save the space. The results are available upon request.

Performance Evaluations under Different Maintenance Costs and Policies

We would like to point out that unavailability of true labels is the case in several real applications. In such application domains, it is common, to construct labels based on operators and expert knowledge. As an example, we can refer to the pioneering research work of Reference [Jardine et al., 1998], which develops a replacement maintenance policy based on PHM using the real dataset from a fleet of heavy hauler truck transmission's spectrometric oil analysis. In order to calculate the required cost function, the stochastic behavior of metal particles (covariates), which are represented by ppm of wear metal particles in engine oil, should be defined. As true labels are not available, authors assumed that covariates behave as a homogeneous Markov process and determined the corresponding transition probability matrix. To determine the transition probability

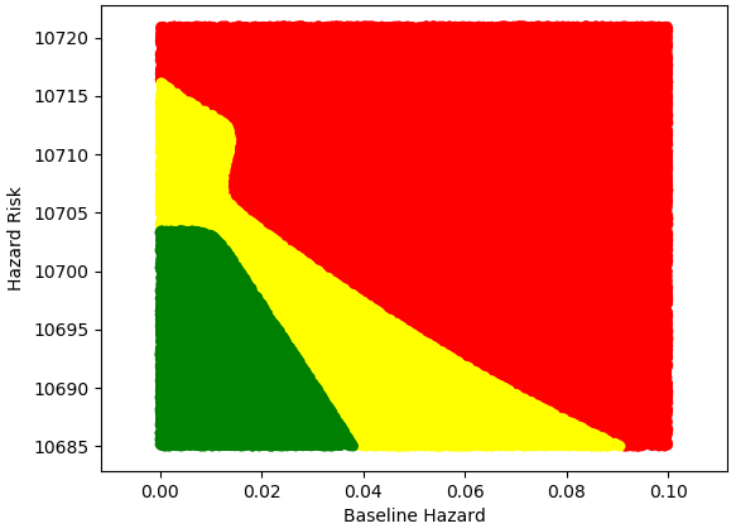
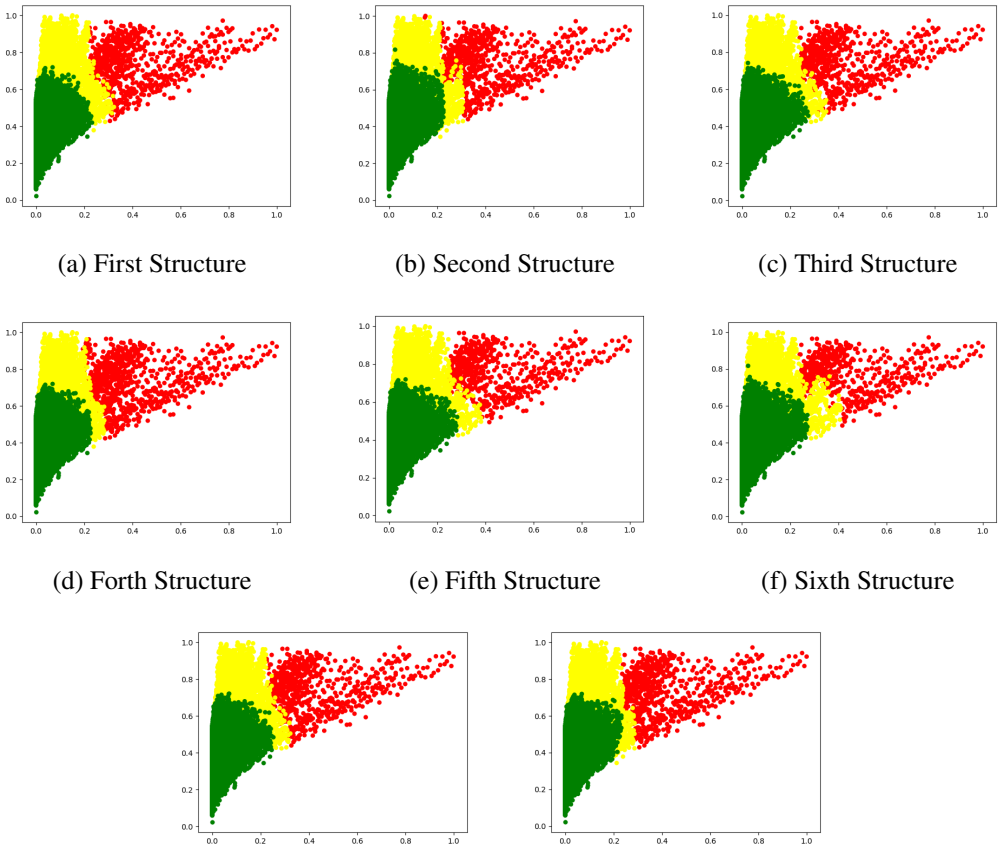


Figure 3.14: Optimizing the CBM decision based on Feed Forward Neural Network.

matrix for the underlying two covariates, i.e., iron and copper, the value of metal iron and copper are divided into 5 intervals, representing the states of the Markov process. In particular, [0,15) interval for iron (ppm accumulated) is considered as State 0, i.e., healthy state; [15, 30) is considered as State 2 and so on. The classification of states and labeling of the CM data, was selected based on expert knowledge.

In the case of the C-MAPSS data, we are dealing with a situation similar to the above discussion. Our main objective for introducing the aggregated clustering method is to address the issue of dependence on expert knowledge. In this regard, we have developed a ML-based solution for construction of labels without human intervention. In order to validate our proposed model, we evaluate the performance of the model by considering different maintenance policies. It is worth mentioning that, to the best of our knowledge, this is the first work on the C-MAPPS data developed within the prognostic and health management context as such we cannot directly compare our results with other methods. Therefore, to validate the proposed model, we evaluate its performance under different maintenance costs and policies by considering cost minimization as the objective function based on the newly developed RL model. In particular, performance evaluations are performed under the following six different scenarios:

(1) **Scenario 1**: Replace only on failure (R-O-O-F) policy with failure cost of 40, i.e., $C_F = 100$.



(a) First Structure (b) Second Structure (c) Third Structure

(d) Forth Structure (e) Fifth Structure (f) Sixth Structure

(g) Seventh Structure (h) Eighth Structure

Figure 3.15: LSTM decision graphs.

(2) **Scenario 2**: R-O-O-F policy with failure cost of 40, i.e., $C_F = 40$.

(3) **Scenario 3**: R-O-O-F policy with failure cost of 20, i.e., $C_F = 25$.

(4) **Scenario 4**: Perform minor and major repairs with associated costs of $C_{PM} = 20$ and $C_F = 40$, respectively.

(5) **Scenario 5**: Perform minor and major repairs with associated costs of $C_{PM} = 20$ and $C_F = 25$.

(6) **Scenario 6**: Perform minor and major repairs with associated costs of $C_{PM} = 20$ and $C_F = 100$.

Results are shown in Table 3.6. The following conclusions can be obtained form results: (i) As the results show, preventive maintenance strategy is the optimal maintenance policy in comparison to

Table 3.6: Evaluation results obtained under different maintenance cost and policy scenarios.

Scenarios	Total Cost	Total Actions	Minor Action	Major Action
1	98.75	399.0125	0	0.98
2	45.5	398.8625	0	1.13
3	23.125	399.075	0	0.925
4	40.75	398.7125	0.5375	0.75
5	28.6875	398.725	0.6375	0.6375
6	85.5	398.25	0.775	0.7

the R-O-O-F policy. It makes sense as the hazard rate of the system is increasing over time; (ii) By increasing the failure cost, the total cost will be increased accordingly; (iii) The average number of minor actions in Scenario 4 is less than that of Scenario 6. We were expecting this behavior form the proposed model. The logic behind this is that when the failure cost is increasing while minor maintenance cost is fixed, the system avoids to go to the failure state (due to high failure cost) and it prefers to perform more minor actions, and; (iv) When the cost of minor and major actions are close, the system will either do major or minor action which is logical, i.e., Scenario 5.

Extension to Multi-Unit Systems

The proposed model can be modified for application to a system consisting of heterogeneous functional elements (referred to as a multi-unit system). Consider a multi-unit system (such as the power supply system) with n components. Each component can have different numbers of sensors, i.e., the i^{th} unit can have n_i number of sensors that measure specific covariates that can indicate the health of that component. Afterwards, we can fit n different clustering models for each component and extract the cluster labels. In the RL section, the state definition should change. We can have two different scenarios and two different approaches to define the states: (i) The cluster labels can be aggregated like what we have done in the third step of our clustering procedure. In this case, the state-space has one dimension, and; (ii) We can avoid doing the aggregation and use each label independently. In this case, our state-space has n dimensions. The action-space would also change, where the number of actions for each component depends on how many different actions are applicable. The final action space is a collection of all possible combinations of localized actions.

For example, consider a system with two units ($n = 2$). The first unit with 10 sensors, and

the second one with 20 sensors. First, we should fit two different k-means clustering models on the sensory data from the first unit and the second unit, and the cluster labels from each clustering model are extracted. We call these labels S_1 and S_2, with ($S_1, S_2 \in 0, 1, 2$), respectively. In the first scenario for defining the RL state-space, the aggregation can be done as follows:

$$S_{\text{Total}} = \begin{cases} 0 & S_1 = 0 \ \& \ C_2 = 0 \\ 1 & S_1 = 0 \ \& \ C_2 = 1 \\ 2 & S_1 = 0 \ \& \ C_2 = 2 \\ 3 & S_1 = 1 \ \& \ C_2 = 0 \\ 4 & S_1 = 1 \ \& \ C_2 = 1 \\ 5 & S_1 = 1 \ \& \ C_2 = 2 \\ 6 & S_1 = 2 \ \& \ C_2 = 0 \\ 7 & S_1 = 2 \ \& \ C_2 = 1 \\ 8 & S_1 = 2 \ \& \ C_2 = 2 \end{cases}$$

It is noteworthy to mention that this type of aggregation is the most strict one since we have considered a different space for each combination, but this could be simplified as well. For example, we can define three states as follows:

$$S_{\text{Total}} = \begin{cases} 0 & S_1 = 0 \ \& \ C_2 = 0 \\ 1 & S_1 \neq C_2 \\ 2 & S_1 = 2 \ \& \ C_2 = 2 \end{cases}$$

If we want to use the second scenario to define the state, the state space is a two-dimensional array, i.e., (S_1, S_2), and there is no need for a global aggregation. Finally, if we consider three actions (i.e., do nothing, minor repair, and major repair), for each unit, we will have 9 different combinations that constitute our final action-space. As a final note, we should elaborate further application of the proposed framework to a heterogeneous multi-state system. In such scenarios, either the variability between the sensors' measurements could be high, or the type of some of the sensors' measurements could differ from each other. Since the scale of the data that each sensor can measure differs from each other, high variability could be the case most of the time in the maintenance. Feature scaling

is a crucial part of the pre-processing step that takes care of the noted problem. Data normalization and standardization are two effective but simple feature scaling techniques that map the original data space into a new space in which all the features have the same scale. Solving the other problem, i.e., different data types, consists of a set of pre-processing techniques prior to using the method proposed in this chapter. These pre-processing techniques should bring all the data into the same domain (format). In summary, the proposed method can be applied to a heterogeneous system by adoption of the aforementioned pre-processing techniques.

3.4 Summary

While RUL prediction is the main focus of ML modeling based on the NASA's C-MAPSS dataset, we believe that RUL alone without design of a proper decision support system is not sufficient for maintenance decision-making. The chapter aimed to address this gap and proposed an efficient and novel hybrid MDSS framework for fault diagnostic and prognostic considering CM data. The proposed MDSS model is a hybrid ML-based solution coupled with statistical techniques. We consider the MDSS to have three distinguishable states, namely, healthy, warning, and failure state. To find an optimal maintenance policy, we concentrate the attention on a time-dependent PHM augmented with a semi-supervised ML approach. The MDSS monitors the hazard rate of the system based decision graphs to continuously detect any abnormality in data that may indicate a change in the process or the system. Given that sensors' measurements in the C-MAPSS dataset do not have decision labels, an intuitively pleasing unsupervised procedure is developed to extract and build the required labels. In order to design such a smart system, different prediction models are constructed, each with a unified decision graph. Several different ML models are utilized and evaluated ranging from traditional ML models to Ensemble Learning Models and DL-based architectures. The effectiveness of the proposed MDSS model is demonstrated through comprehensive set of comparisons with different ML algorithms.

Chapter 4

HazNet: Non Linear and Deep Hazard Architecture for Maintenance Management

The hazard rate is one of the frequently used measures in maintenance that can help prevent the system's failure, obtain an optimal maintenance policy, and develop an MDSS. Conventionally, researchers have been using Cox's PHM to estimate the hazard rate of a system. This traditional model assumes a linear relationship between the sensors' measurements that might be the superficial and shallow assumption for real-world usage. Moreover, to use time-dependent features in the model, more adjustments are required. In addition to this limitation, with the emergence of extensive CM data, statistical models cannot analyze these data. On the other hand, there has been a surge of interest in applying ML/DL methods in the maintenance domain due to their great potential to make the most accurate predictions from a large volume of data and discover hidden patterns and trends in the data that might not appear in the first look. Hence, this chapter proposes a non-linear, time-dependent, and DL-based variant of the Cox's PHM. Therefore, this chapter proposes a non-linear, time-dependent, and DL-based variant of the Cox's PHM. The proposed approach can learn the hazard risk from unlabeled data without requiring domain expertise and/or feature extraction for

the training process. Different structures of the proposed model are trained and compared. A well-known evaluation metric in the survival analysis called the concordance index is used to compare the trained models. The proposed model shows promising results and great potential to be used in the maintenance domain.

4.1 Problem Statement

As discussed previously, although the the results regarding the RUL predictions are promising and highly accurate, RUL cannot be considered an adequate and efficient measure for maintenance decision-making in some systems. For example, in a system with catastrophic failure consequences, it is preferable to estimate how much a machine or a system is probable to operate properly without failure until the next inspection interval [Jardine et al., 2006]. For such a system, the remaining time before failure, i.e., RUL, is not a reliable measure for maintenance operators to decide whether the current inspection interval is suitable or not. With that in mind, the hazard rate could be both an efficient indicator that can address the noted problem properly and can be used to calculate RUL. Hazard rate has been used in different research areas such as maintenance, healthcare, and many more. As already mentioned, Cox's PHM was first introduced by Cox [Cox, 1972,7] to calculate the hazard rate. This model estimates the hazard rate of a system considering both the age of a system and the system degradation. Covariates are the sensors' measurements or features that associate the system degradation with the hazard rate. Because of its simplicity yet general usability, it is one of the most frequently used models in studying the importance of age and different covariates on the system, calculating the RUL, and many more. For example, a new approach called Cox Proportional Hazard Deep Learning (CoxPHDL) is developed in [Chen et al., 2020] to tackle the data sparsity and censoring problem. Authors used Cox's PHM to estimate the Time Before Failure (TBF) of the censored data. These estimations are considered as labels, and an LSTM network is established to predict the estimated TBF. In [Zhao et al., 2018], Cox's PHM is used as a part of a health assessment methodology to describe the hazard rate of the time to failure for a deteriorating system subject to CM data. A novel CBM policy with two sampling intervals is proposed in [Naderkhani et al., 2017]. In their policy, a longer interval is used for the new or renewed system where a shorter interval is

used whenever the estimated hazard rate reaches a certain threshold. It is assumed that the system is subject to degradation that is later described by Cox's PHM.

However, to the best of our knowledge, a linear relationship between the covariates, i.e., linear hazard risk, is considered in most cases in the maintenance research area. Assuming a linear hazard risk may be a superficial assumption and not applicable in practical settings. To resolve this issue, one can consider high-level interactions between the covariates. However, this could be computationally expensive as the number of covariates grows. Therefore, researchers in survival analysis and healthcare areas have developed non-parametric models to estimate the hazard risk. One of the best and premier papers in this regard is proposed in [Katzman et al., 2018]. The authors developed a nonlinear extension of the Cox's PHM known as DeepSurv and used a feed-forward neural network to estimate the hazard risk without any human interference. This model, however, may not be used properly in the maintenance domain since the nature of maintenance data is different from healthcare data. That is because in the maintenance context since we are monitoring the system through time, most of the time, data is time-dependent. For time series prediction feed-forward neural network cannot be used, and other types of DL-based models, such as RNN or CNN, should be considered. Moreover, their model is only able to learn from the failure or observed event data because of the way that their loss function is formulated. In this way, they are losing lots of useful information that censored data may contain. Finally, since the proportion of the failure data is usually less among the whole dataset in the maintenance context, we may not be able to use a deep and complex model since their network cannot learn properly.

Based on the above mentioned motivations, and to address the aforementioned problems, we propose a non-linear, time-dependent, and DL-based variant of the Cox's PHM. The proposed model shows promising results and great potential to be used in the maintenance domain. The reminder of the chapter is organized as follows: Section 4.2 provides a description of the dataset used in this case study. Section 4.3, proposed our method. Results are discussed in Section 4.4. Finally, Section 4.5 concludes the chapter.

4.2 Dataset Description

To evaluate the proposed model, we used the same dataset as the last chapter known as Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) provided by NASA [Saxena et al., 2008]. As stated in Chapter 3, this dataset consists of four subsets of data known as FD001, FD002, FD003, and FD004. Each dataset has 21 sensors' measurements and three operating settings values. In this case study, we have used the FD001 subset of data in which there are 100 engines under operation and subject to random failure. More discussion on this dataset can be found in Section 3.2.

4.3 Proposed Model

Previously, Cox proposed a partial likelihood without involving the baseline hazard, and suggested that by maximizing the partial likelihood, we can estimate the coefficients [Cox, 1972,7]. The aforementioned partial likelihood is given by

$$PL(B) = \prod_{i=1}^{n} \left(\frac{exp\left(B^T x_i\right)}{\sum\limits_{j \in R_{T_i}} exp\left(B^T x_j\right)} \right)^{E_i} , \tag{29}$$

where $R_t = \{i : T_i > t\}$ denotes the risk set, which is all the individual at risk at time t. Reference [Katzman et al., 2018] used the average negative log partial likelihood of Eq. (29) as the loss function to train the network, i.e.,

$$Loss(\theta) = -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\hat{h_\theta}(x_i) - \log \sum_{j \in R_{T_i}} e^{\hat{h_\theta}(x_j)} \right) , \tag{30}$$

where $\hat{h_\theta}(x_i)$ is the output of the network, which is a single node with a linear activation function. However, the noted loss function has the following two problems:

- First, since the summation is over the observed events ($i : E_i = 1$), censored data do not contribute to the loss computation in each epoch. Therefore, the model cannot learn from the censored data in the training process.

- Second, when there are less failure data, the loss function cannot be used to train a deeper

neural network that depends on a considerable amount of data. This might be the case most of the time in the maintenance domain since we are observing the system over time until a failure happens. Failure could be costly and we cannot afford to run the system several times until failure happens. Therefore, we only have few failure data points (observed events) compared with abundance of censored data when it comes to maintenance.

These issues will be addressed in the following subsections.

4.3.1 Learning from Censored Data

E can be seen as a weight in loss computation that decides whether a data point should fully contribute to the computation or not at all. Therefore, the model can only learn from the failure data. This might not be preferable since in many cases, less amount of failure data is available, and this might prevent us from using more complex and cutting-edge DL models since these models require a substantial amount of data. Moreover, although censored data may not be as enriching as the failure data, they still contain useful information that we can use to train a more accurate and reliable model. For example, the observation right before the system failure is considered censored data, but it might still provide valuable information. To address these problems, we came up with the idea of considering E as a continuous random variable rather than a discrete random variable, which is either equal to 0 or 1. Therefore, rather than assuming $E_{i,j} = 0$ for every censored data, $E_{i,j}$ for the i^{th} observation of the j^{th}th subject is calculated as follows:

$$E_{i,j} = \frac{T_{i,j}}{\max\limits_{i}\{T_{i,j}\}} \ . \tag{31}$$

In this way, the contribution of each data point in the loss function computation will be based on the time of the observation, $T_{i,j}$, and the failure time of the i^{th} run, $\max\limits_{i}\{T_{i,j}\}$. Eq. (31) suggests that those censored observations that are closer to the failure time would contribute more to computing the loss function. This is a logical and justifiable assumption that the censored data near failure time might contain more useful information, and as a result, they should contribute more to the computation. In this regard, we will not lose useful information that censored data might contain, and it allows us to train more complex models because we have more data.

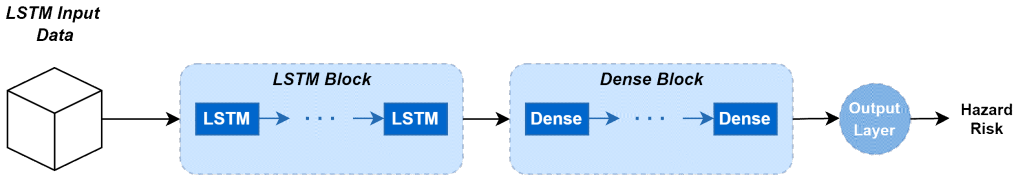


Figure 4.1: The proposed LSTM network architecture. Stacked Dense layers can be used after LSTM cells to make the network deeper.

In this case study, we used two popular DL models for time series analysis, namely LSTM and one-dimensional CNN as described in Subsection 2.2.1. These are among the widely used models for time series analysis due to their prediction power and general usability in many fields and contexts. Hybrid DL-based models can be also achieved by combining different kinds of DL models. These models will enable us to take advantage of using various models simultaneously. Consequently, it may help us achieve better results. Since both LSTM and 1D-CNN are two DL models which can be used for time series analysis, we can integrate these single models into one hybrid model. We will discuss the proposed single and hybrid models in the next subsection.

4.3.2 Proposed Model Framework

We use modern DL models to fit the data. There are many noted models in the literature, but some common models for time series analysis are LSTM and 1D-CNN that we described in 2.2.1. The same 1D-CNN network architecture as shown in Fig. 2.2 is used with different hyperparameters, including the number of convolution and Dense layers, the number of nodes per Dense layer, the number of filters per convolution layer, and the activation function in each layer.

For the LSTM-based model, we can also stack dense layers after the LSTM cells as shown in Fig. 4.1 to make the network deeper. Different network configurations were used to tune the hyperparameters, including the number of LSTM and Dense layers, the number of nodes per LSTM and Dense layers, dropout rate, and activation function in each layer.

Another interesting DL-based model to consider is a hybrid DL-based model. As we discussed in 4.3.1, hybrid models combine different DL models to take advantage of the power of different models. These models are more complex than the single models and require more data to be trained. By introducing a continuous event variable and being able to use censored data in the model training,

now, we can also consider a hybrid model as well.

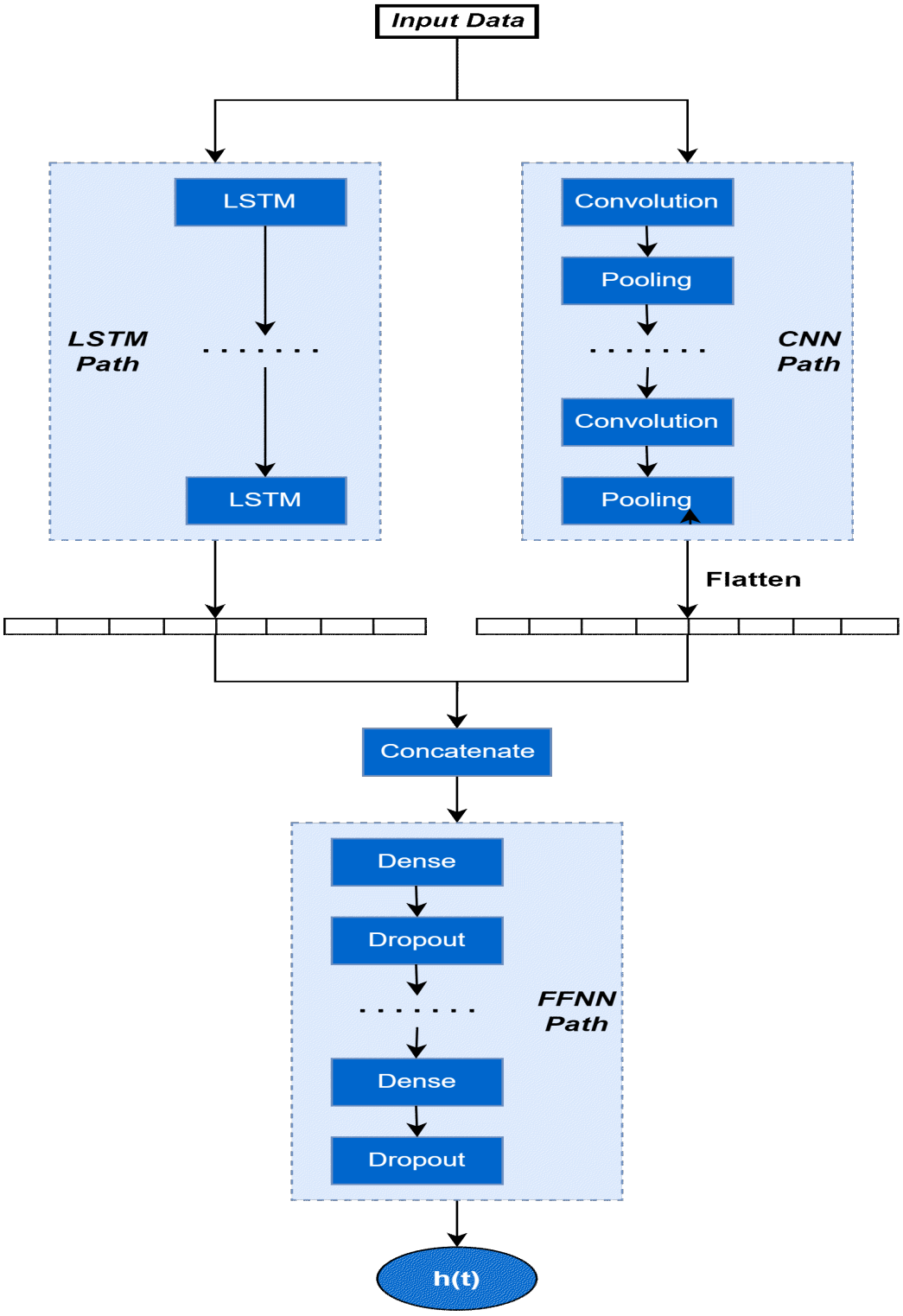


Figure 4.2: The proposed hybrid model architecture combines LSTM and 1D-CNN. After concatenating the output of LSTM and 1D-CNN paths, it will be fed into an FFNN network.

As we stated earlier, LSTM and CNN are two frequently used models for time series prediction. A hybrid model can be made based on these two models. The architecture of the model is shown in Fig. 4.2. The same input data was fed into LSTM and CNN paths. After flattening the CNN path result, it was concatenated with the LSTM path result. Then, the concatenated data was fed into the feedforward path containing Dense layers. Dropout layers were used to prevent overfitting. The same loss function and same optimizer as single models were used to train the model.

We have used 200 epochs and the Nadam optimizer to train all the models. Nadam optimizer is an extension of the well-known Adam optimizer that adopts Nesterov momentum. Nadam optimizer allows faster convergence and improves the quality of the learning [Dozat, 2016]. Two different learning rate values, 0.001 and 0.005, were used to train the LSTM models, and the 0.001 learning rate was used to train the 1D-CNN and the hybrid models. Moreover, l_2 regularizer equals 16, and Dropout layers are added to all the trained models to prevent overfitting. Different dropout rates were used for LSTM and hybrid model but, a 0.2 dropout rate was considered for the 1D-CNN model. All models are formulated and trained using TensorFlow in Python. Next, we present our experimental results to evaluate performance of the proposed framework.

4.4 Experimental Results

As stated previously, we defined the loss function based on a continuous event variable and the data is prepared accordingly to be processed by three different DL models that we mentioned in Section 4.3.2.

After shuffling the 100 engine numbers, 20 engines and their associated data were chosen for the test set and the rest of the data for the training set. The sensors' measurements were also normalized and rescaled into the range of $[0, 1]$ so they can contribute equally to the model training. After these preprocessing and preparing steps, data was ready to be fed into our network.

We trained these models with different hyperparameters and used 5-fold cross-validation to evaluate them reliably and robustly. It is also noteworthy to mention that 10-fold cross-validation could also be used. However, since different hyperparameters settings were considered, it was time-consuming to get the 10-fold cross-validated results.

The K-fold cross-validation splits the dataset into K equal parts. However, if we followed the conventional procedure of cross-validation, each engine's data would end up in different folds. Therefore, regarding the nature of the maintenance data, we changed this procedure slightly. Hence, instead of splitting the observations data into five equal folds, without considering from which engine these observations are, we first shuffled the engine's number randomly and then, divided the engines into five folds. This way, we made sure that in each fold, there are data of 20 engines. Each network structure is trained using both discrete and continuous event detectors to further compare their results as well.

The C-index results in Table 4.1 show that the proposed model could successfully learn from time series data to predict the hazard risk. It is worth mentioning that a model with C-index above 0.7 is considered well performed. Using the continuous event variable also yields close performance to the discrete event variable, and sometimes it outperforms that. The 1D-CNN results in Table 4.2 suggest that in general, considering continuous event detectors yields better results. Apart from the fact that discrete event detectors had less accurate results in most cases in comparison with the continuous event detectors, the stability of the model predictions has also decreased. The result of using the hybrid model described in Section 4.3.2 is shown in Table 4.3. The discrete event variable performed better than the continuous variable in this case. Moreover, considering both the average and STD of the cross-validated results, the best model was achieved using the hybrid model and discrete variable. For easier comparison, the results are summarized in Fig. 4.3.

Table 4.1: LSTM Results

Event	Dropout Rate	Learning Rate	Activation Function	# Dense - # LSTM Layers	# Nodes per Dense Layer	# Nodes per LSTM Layer	Average of cross-validated C-index	SD of cross-validated C-index
Discrete	0.5	0.001	ReLU	1-1	16	8	**0.906**	0.036
	0.5	0.001	ReLU	1-1	32	16	0.878	0.047
	0.5	0.005	SeLU	2-2	32, 16	16, 32	0.851	0.041
	0.5	0.001	SeLU	4-4	16, 32, 32, 16	16, 32, 32,16	0.884	**0.015**
	0.25	0.001	ReLU	4-2	16, 32, 32,16	32, 16	0.837	0.074
Continuous	0.5	0.001	ReLU	1-1	16	8	0.864	0.044
	0.5	0.001	ReLU	1-1	32	16	**0.895**	0.017
	0.5	0.005	SeLU	2-2	32, 16	16, 32	0.877	**0.01**
	0.5	0.001	SeLU	4-4	16, 32, 32, 16	16, 32, 32,16	0.884	0.038
	0.25	0.001	ReLU	4-2	16, 32, 32,16	32, 16	0.871	0.034

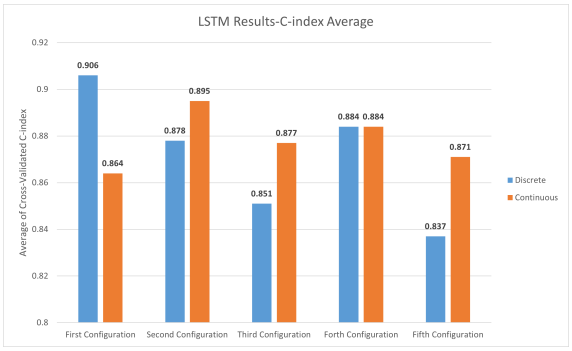
Table 4.2: 1D-CNN Results

Event	Activation Function	# Dense - # Conv Layers	# Nodes per Dense Layer	# Filters in each Conv Layer	Average of cross-validated C-index	SD of cross-validated C-index
Discrete	Tanh	1-1	64	32	0.588	0.098
	SeLU	1-1	64	32	0.645	0.155
	ReLU	4-2	16, 32, 32, 16	16, 32	**0.902**	**0.036**
	SeLU	3-2	32, 64, 32	16, 32	0.839	0.097
Continuous	Tanh	1-1	64	32	0.789	0.026
	SeLU	1-1	64	32	0.846	0.045
	ReLU	4-2	16, 32, 32, 16	16, 32	0.866	**0.016**
	SeLU	3-2	32, 64, 32	16, 32	**0.873**	0.027

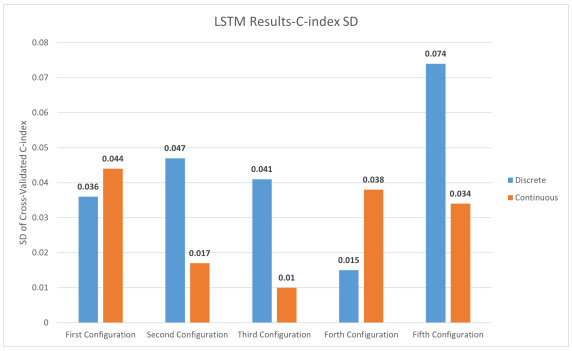
Table 4.3: Hybrid Model Results

Event	Dropout Rate	Conv-LSTM-Dense Activation Function	# Conv-# LSTM-# Dense Layers	# Filters in each Conv Layer	# Nodes per LSTM Layer	# Nodes per Dense Layer	Average of cross-validated C-index	SD of cross-validated C-index
Discrete	0.5	ReLU-ReLU-ReLU	2-1-3	16, 32	16	32, 64, 32	0.89	0.035
	0.2	SeLU-SeLU-ReLU	2-2-4	16, 32	16, 32	16, 32, 32, 16	0.897	0.05
	0.2	SeLU-SeLU-ReLU	2-4-3	16, 32	16, 32, 32, 16	32, 64,32	**0.905**	**0.025**
	0.5	ReLU-SeLU-ReLU	2-2-3	32, 32	16, 32	16, 32	0.773	0.115
	0.5	ReLU-SeLU-ReLU	2-2-3	32, 64	16, 32	16, 32	0.89	0.032
Continuous	0.5	ReLU-ReLU-ReLU	2-1-3	16, 32	16	32, 64, 32	0.864	0.049
	0.2	SeLU-SeLU-ReLU	2-2-4	16, 32	16, 32	16, 32, 32, 16	**0.879**	**0.02**
	0.2	SeLU-SeLU-ReLU	2-4-3	16, 32	16, 32, 32, 16	32, 64,32	0.814	0.051
	0.5	ReLU-SeLU-ReLU	2-2-3	32, 32	16, 32	16, 32	0.865	0.026
	0.5	ReLU-SeLU-ReLU	2-2-3	32, 64	16, 32	16, 32	0.823	0.036

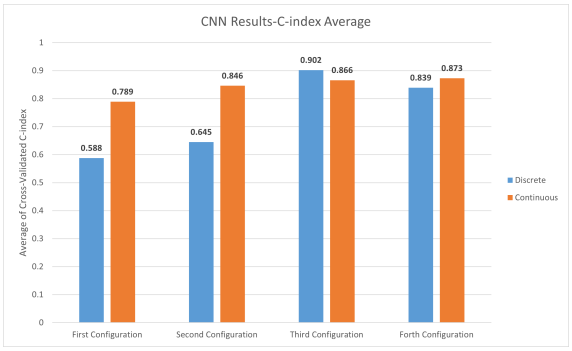
One interesting point is that the continuous event detector has a more stable and better standard deviation in many cases. This shows that when the network structure gets more complex, using the continuous event variable tends to perform more steadily. However, even though the continuous event variable yielded better results in some cases, we got the best results using the hybrid model and discrete event variable. This could be due to the fact that when we have an abundant number of censored data, some of them might not be useful for the model training. Hence, when we are using the discrete event variable, the training of the model will be solely based on the failure data. Whereas using the continuous event variable participates the censored data in the training process and therefore, it might decrease the importance of the failure data that the training of the model was based on previously. This might be a drawback of using a continuous event variable when we have



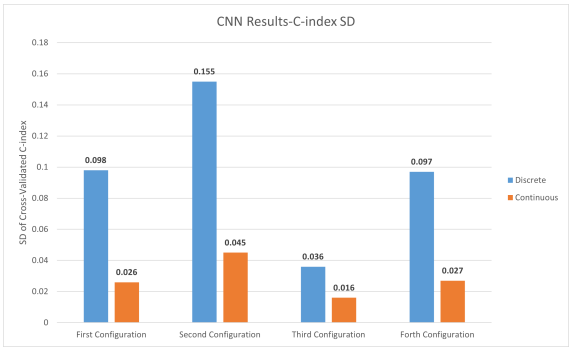
(a) LSTM-Average of C-index



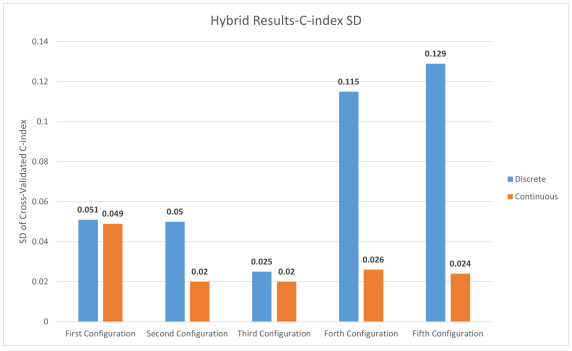
(b) LSTM-SD of C-index



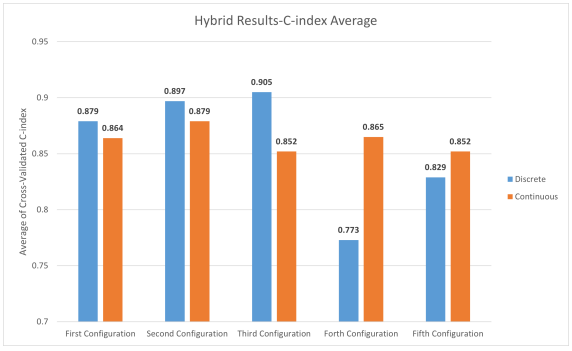
(c) 1D-CNN-Average of C-index



(d) 1D-CNN-SD of C-index



(e) Hybrid Model-Average of C-index



(f) Hybrid Model-SD of C-index

Figure 4.3: The summary of the average and standard deviation of the cross-validated results using LSTM, 1D-CNN, and the hybrid model for discrete and continuous event variables

enriching failure data that could relate the system failure with the sensors' measurements when the failure has happened. However, this might not be the case when such sensors' measurements are not available, for example when we have image data, and more computation is needed to extract features. Finally, our general goal here was to demonstrate that considering a continuous event variable has the potential to be used in real practices and yields close results as to the discrete variable. It is suggested that both continuous and discrete variables be used and then decide which

one performs better based on the application.

For illustration purposes, the first, the second, and the twenty-fifth engines are selected to plot their hazard risk predictions. The plots are shown in Fig. 4.4. The model used for prediction here has the following structure:

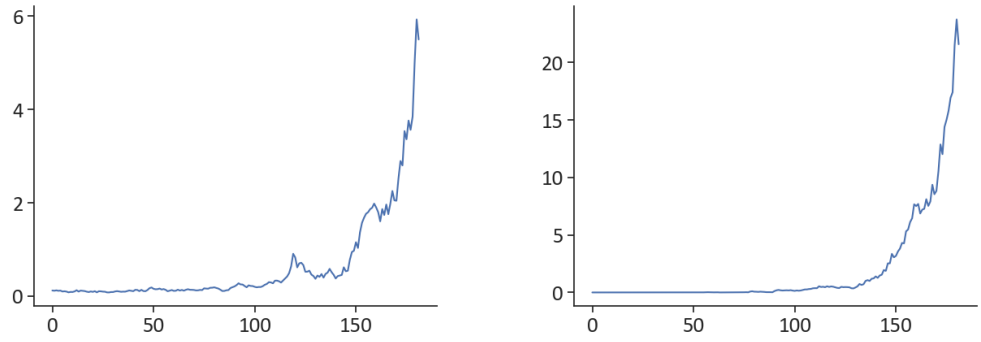
Table 4.4: Network structure used for illustration purposes

Dropout Rate	0.5
Learninge Rate	0.001
Activation Function	ReLU
# LSTM - # Dense Layers	1-1
Nodes per LSTM Layer	8
Nodes per Dense Layer	16

The cross-validated result of the above model is reported in Table 4.1 in the first and the sixth rows for discrete and continuous event detectors, respectively. These figures demonstrate that the first engine failed earlier. At the same time slot, it can be seen that it also has a greater hazard risk in comparison with the second and the twenty-fifth engines that failed later. This is precisely aligned with how C-index works and what we were looking for. Moreover, it shows that the hazard risk predictions using both continuous and discrete event detectors are increasing over time which was expected to be.

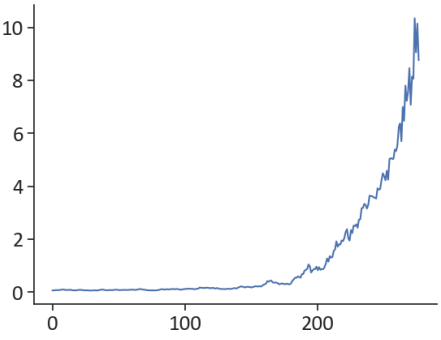
4.5 Summary

Although Cox's PHM is a frequently used model in the literature to estimate the hazard rate of a system, the linear relationship between the covariates and time independency of the covariates might be a superficial assumption and not applicable in many real-world problems. This chapter aimed to propose a non-linear and DL-based variant of the Cox's PHM to relax these assumptions. Moreover, considering the high cost of failure data gathering and scarcity of such data in comparison with the censored data, the developed model is able to learn from the censored data as well. This would suggest a great potential to use deeper and more complex DL-based models in the maintenance domain for PHM purposes. To evaluate the proposed model, C-MAPSS data is used. Two different DL single models, known as 1D-CNN and LSTM, alongside a hybrid model that could combine the power

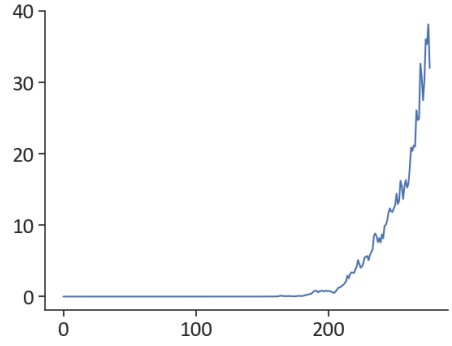


(a) 1st Engine-Continuous event detector

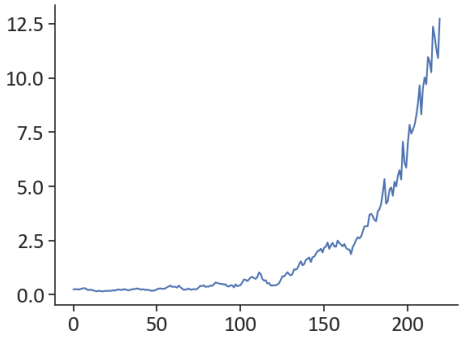
(b) 1st Engine-Discrete event detector



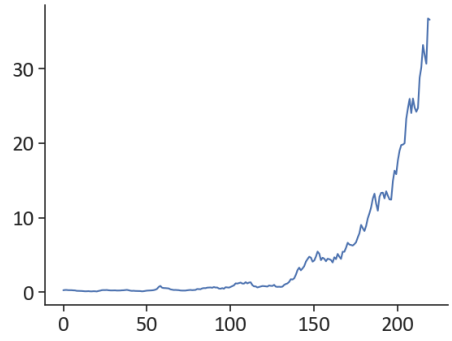
(c) 2nd Engine-Continuous event detector



(d) 2nd Engine-Discrete event detector



(e) 25th Engine-Continuous event detector



(f) 25th Engine-Discrete event detector

Figure 4.4: Predicted hazard risk for different engines

of these two single models are used. For better hyperparameters tunning, different configurations of the developed models and K-fold cross-validation are used.

Chapter 5

Deep Learning-based Survival Analysis for a System Subject to Stochastic Degradation using High-Dimensional Condition Monitoring Data

As discussed in the last chapter, prognostic and health management is one of the important and common methods to improve assets' reliability, safety, and maintainability by decreasing the occurrence of failure events. CBM is a state-of-the-art maintenance policy that utilizes CM data to track the health of the system. With new advancements in IIoT and data gathering tools, like smart sensors, nowadays, the collection of a vast amount of real-time data is possible. This provides a new opportunity to use an efficient CBM model using these high-quality real-time data. These data could come from a variety of devices such as temperature, force, and vibration sensors or imaging devices. Among these devices, dealing with imaging devices could be harder than the others. That could be due to the fact that images are high-dimensional data, and classic methods like statistical models cannot handle them. Yet, images provide a rich amount of information and are easy to implement in the real world. The novel method proposed in the last chapter requires signals or sensors' measurements as input data, and it cannot use images. Hence, in this chapter, we will discuss a new

method to predict the hazard risk using image data. The general goal here is to show an application of the proposed method in the last chapter when the input data is changed. Therefore, we are willing to show that we can utilize the developed model in various settings and for different data types without any general change or modification. The proposed method is based LSTM and Dense layers. Several models with different hyperparameters settings are tested on a simulated dataset discussed in [Aydemir and Paynabar, 2019] for performance comparison and hyperparameter tunning purposes. A general description of the dataset used in this chapter is provided in Section 5.1. The proposed model and two problems that can be encountered in this setting will be introduced in Section 5.2. Finally, the results will be provided and discussed in Section 5.3.

5.1 Dataset Description

The dataset used in this chapter includes infrared images with 40×20 pixels obtained from rotating machinery presented in [Fang et al., 2019]. Test bearings were run from a healthy state until failure happened. Vibration sensors were used to track the amplitude of the vibration frequency, and failure happened once it crossed a predefined threshold. Four different experiments were run, and the degradation-based images were recorded every 10 seconds. Afterward, more degradation image streams were generated by resampling the stored data obtained from the previous four experiments. In total, the final dataset contains 284 image data streams. A brief statistical overview of the failure time is presented in Table 5.1. An example of an image stream is also shown in Fig. 5.4a.

Table 5.1: Dataset Description

Count	Mean	STD	Min	Quarter 1	Quarter 2	Quarter 3	Max
284	34.032	10.89	16	24	34	43	55

5.2 Model Name

Our study in this chapter aims to predict the hazard risk of a system using images. These images contain information about system degradation. Images differ from sensors' measurement data in the sense that at time t, a sensors' measurement is a 1-D vector containing a row of observation

and a couple of features values. But, when it comes to images, at time t, we have a 2-D array. Moreover, the features are not measured directly, and more computations should be performed to extract such features that could relate to the system degradation. One way of doing that is to use a set of convolutional layers and feed the data to these layers. Finally, the output of these layers will be flattened to a 1-D vector. By doing so, although the model itself will extract useful features, it will introduce more parameters, and the model would be more complex. Another easy yet powerful way of extracting features is to flatten the images in the beginning without using any convolutional layers [Kolouri et al., 2018; Roy and Vijay, 2019; Xu and Duraisamy, 2020]. Consequently, each pixel of the image will be considered as a feature containing information about the degradation of the system. This is a strict way of preparing an image for further analysis but, in this way, each pixel's value will be important for us. As stated previously, our goal here is to show that we can use the same proposed model in Chapter 4 in a new application by modifying the input data instead of changing the whole developed model.

The structure of the proposed model is shown in Fig. 5.1. After flattening the images, we can reshape input data into the required format by LSTM. An Adam optimizer is used to train all the LSTM and Dense layers. The same loss function as described in the (30) in the previous chapter is also used. As we saw in the last chapter, this loss function cannot learn from the censored data. The amount of data that we need depends on the complexity of both the problem and the learning algorithm. Since DL models are complex models and most of the time, little amount of failure data is available in the maintenance area, depending on only failure data and not using the censored data might result in a poorly trained model. Therefore, the same idea, i.e. continuous event detector, as discussed in the previous chapter, is also used here. Models are both trained using continuous and discrete event detectors. A comparison between them is provided in Section 5.3.

As mentioned earlier, the data used in this chapter are images. Images are considered as high-dimensional data. Dealing with such data might be a difficult task, and lower-dimensional data is preferred in some cases. This problem and a method to convert high-dimensional images into a lower dimension are discussed in Subsection 5.2.1.

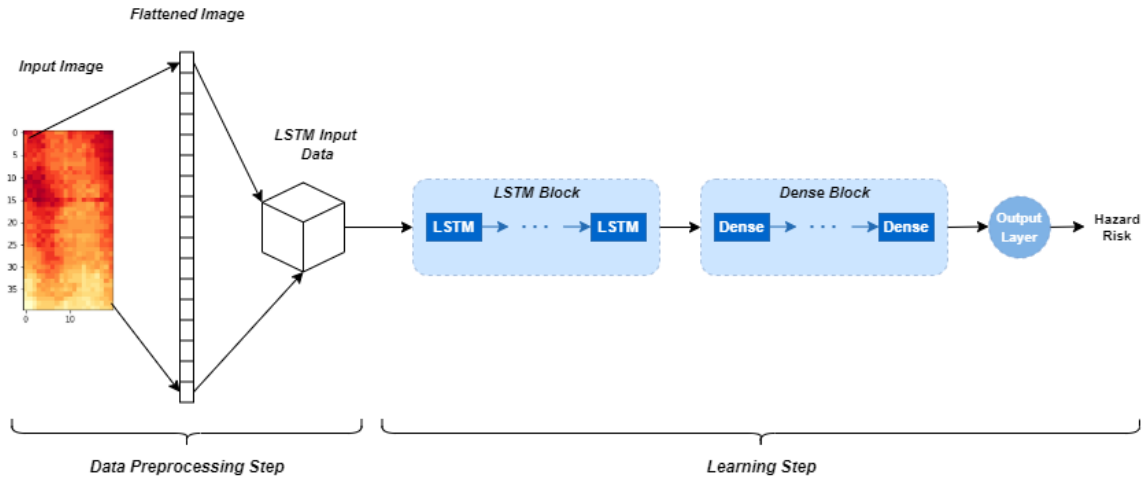


Figure 5.1: Proposed Model Architecture.

5.2.1 Dimension Reduction Technique: Convolutional Autoencoder

As stated earlier, time series predictions using images is a complex task in the DL area. That is mostly because the input data used in these studies are high dimensional, and the time series nature of the data adds more problems. Thus, models used for this purpose are complex and may require a considerable amount of time to be trained. One way to resolve this issue is to use an unsupervised learning method to reduce the input data.

Autoencoder is an unsupervised learning method based on Feedforward neural network that can be used to compress the data into a lower-dimensional space. An autoencoder is composed of two parts: An encoder and a decoder. The input data is passed through the encoder part to produce a lower-dimensional representation of the input, also known as the code or the bottleneck. Then, the code is fed to the decoder part that has the same structure as the encoder part. Decoder tries to build the same input that was fed to the encoder part in the first place. If the decoder can do so, the code or the bottleneck can be considered a lower-dimensional representation of the data.

Since the input data used in this chapter are images, we use a specific type of autoencoder called Convolutional Autoencoder. Convolutional Autoencoders use convolutional layers instead of feedforward layers and as a result, can handle images. An overall representation of this DL model is shown in Fig. 5.2. Like many other DL models, there are some hyperparameters involved in constructing this model as well. The number of filters per convolutional layer, number of layers,

and loss function are some of the hyperparameters. One can attempt to tune these hyperparameters to improve the results. However, in this thesis, we only considered the number of filters in the convolution layers as our hyperparameter. Thus, two different configurations are used as illustrated in Fig. 5.3. Since the number of filters in each convolution layer is the only hyperparameter in this part, we will refer to these two configurations as $4-8-1$ and $16-32-1$ from now on. ReLU activation function is used in convolutional layers, and mean square error was employed as the loss function. Root Mean Squared Propagation (RMSprop) was also used as the optimizer to train the model.

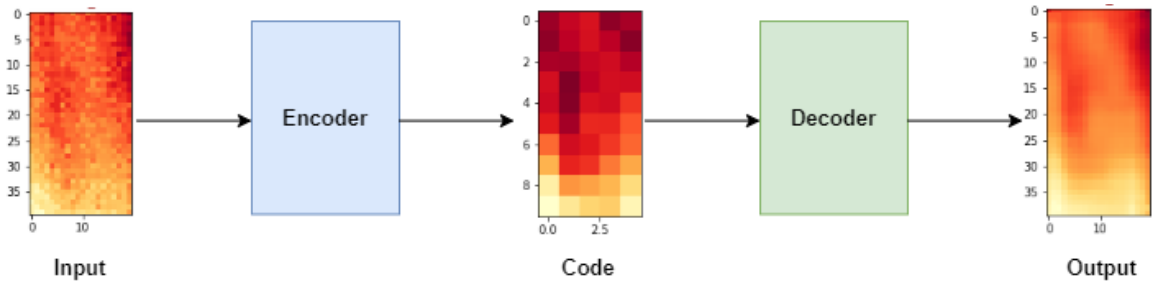
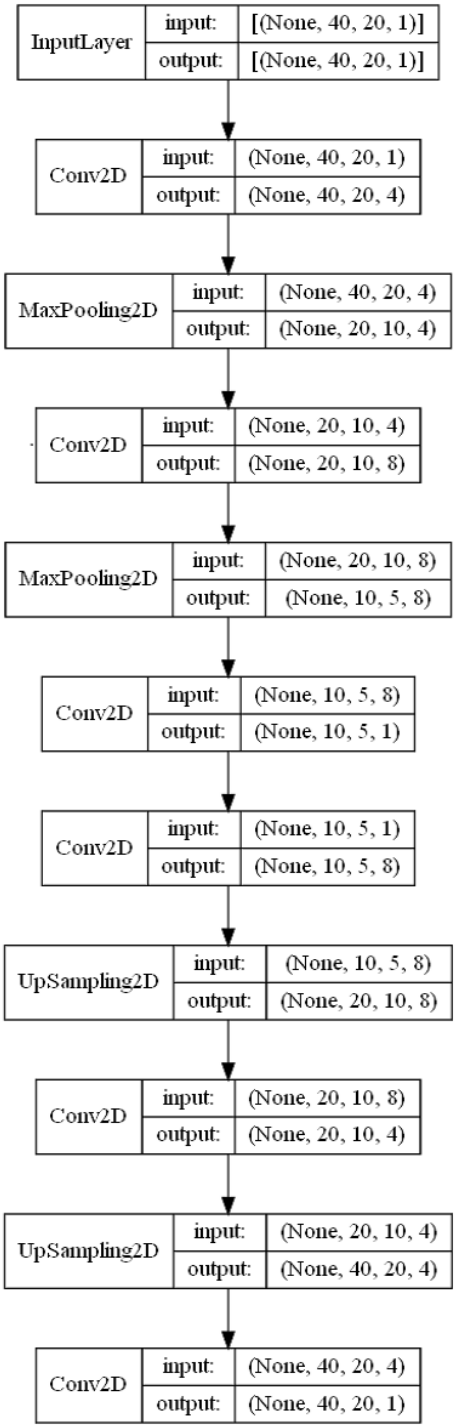


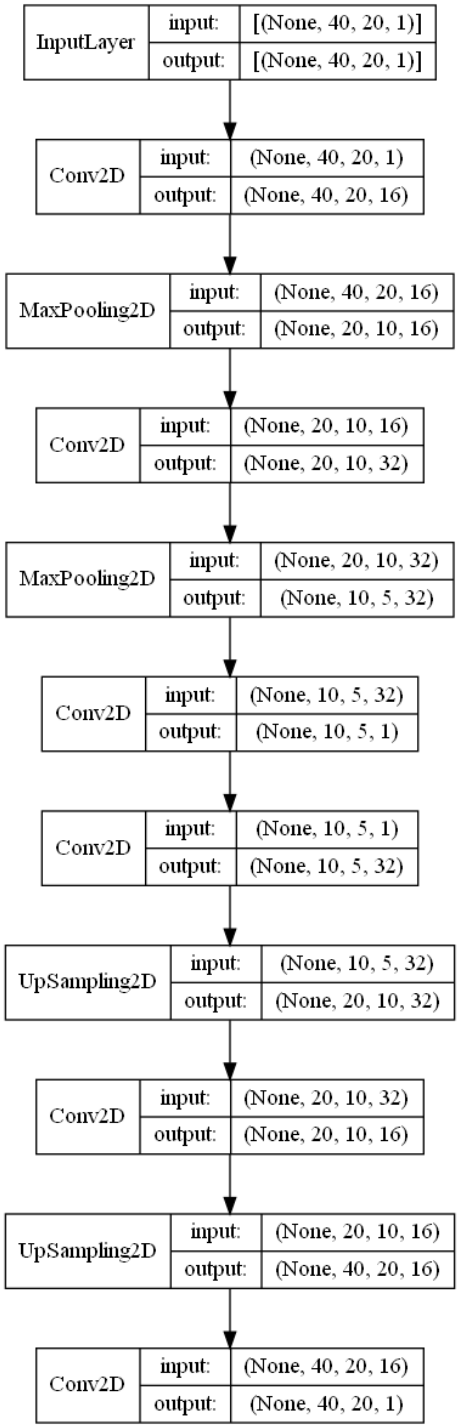
Figure 5.2: Convolutional Autoencoder Architecture.

As it is evident in Fig. 5.3, both will convert the 40×20 pictures into 10×5 images. As a result, if n refers to the number of images in the dataset, we will end up with a $n \times 50$ dataset instead of $n \times 800$, which is a huge reduction. For illustration purposes, an example of an image stream is shown in Fig. 5.4a. Fig. 5.4b and Fig. 5.4c show the reconstructed and compressed images using the first structure, respectively. The same image stream, reconstructed, and compressed images using the second structure are shown in Fig. 5.5. To see the Convolutional Autoencoders performance, both the compressed and the original forms of data will be fed to the hazard risk prediction model, and the results will be compared in Section 5.3.

A noteworthy point to mention here is that as it is apparent in Fig. 5.4, a change in the compressed images over time is evident. In the beginning, the images have a more yellowish color but, over time, we see more red colors. As opposed to this figure, such changes over time in the compressed form of the images are not available in Fig. 5.5. This might suggest that the $4-8-1$ structure would perform better because it shows that it has grasped degradation over time very well.

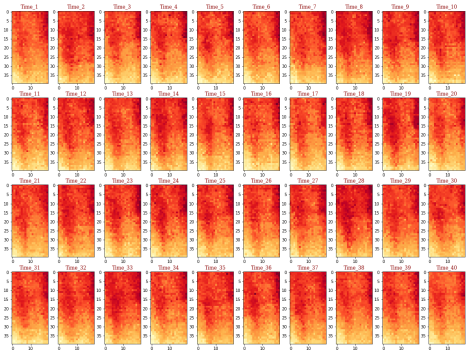


(a) First Structure $(4 - 8 - 1)$

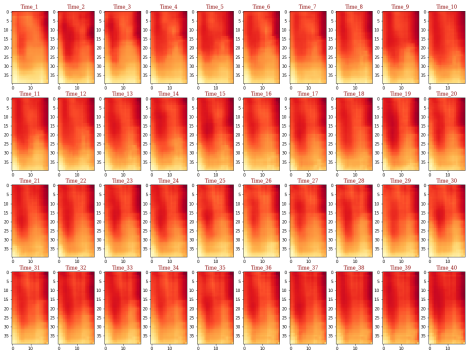


(b) Second Structure$(16 - 32 - 1)$

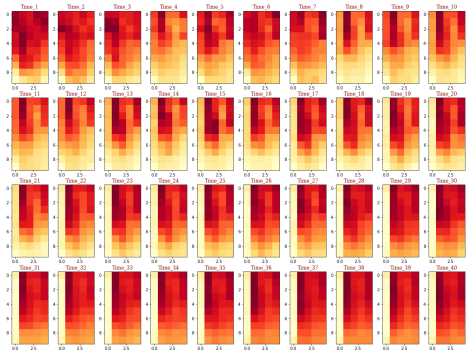
Figure 5.3: Convolutional Autoencoder configurations used in this study.



(a) Original images

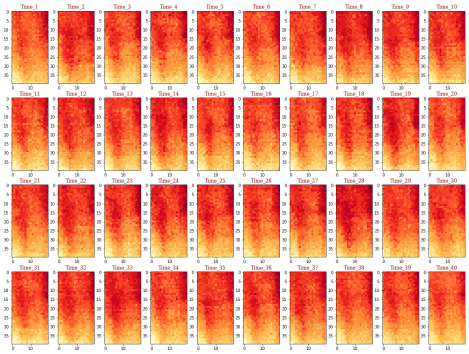


(b) Reconstructed images using the first structure $(4 - 8 - 1)$

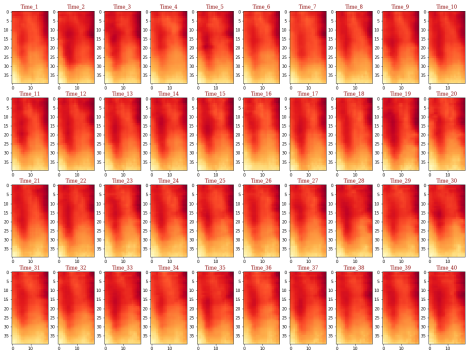


(c) Compressed images using the first structure $(4 - 8 - 1)$

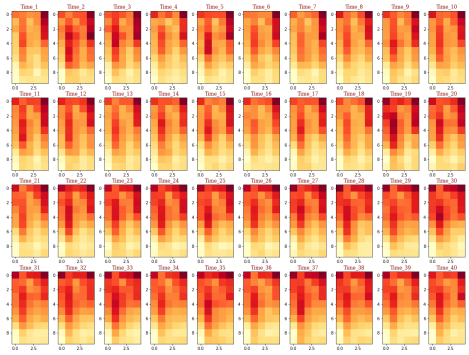
Figure 5.4: Original, reconstructed, and compressed images using the first Convolutional Autoencoder configuration $(4 - 8 - 1)$



(a) Original images



(b) Reconstructed images using the second structure $(16 - 32 - 1)$



(c) Compressed images using the second structure $(16 - 32 - 1)$

Figure 5.5: Original, reconstructed, and compressed images using the second Convolutional Autoencoder configuration $(16 - 32 - 1)$

5.3 Results and Conclusion

As we stated earlier, different models with different structures were trained for hyperparameter tuning purposes. Each model was trained using discrete and continuous event detectors. Adam optimizer with a 0.0001 learning rate was used to train these models. Also, Dropout layer and a l_2 regularization $= 16$ were used to prevent overfitting. A feed-forward fully connected layer consisting of Dense layers could be also used after LSTM layers. Different number of Dense layers, dropout rate, and nodes per layer is also considered. Another important hyperparameter in LSTM models is known as n_past. This hyperparameter indicates how much of the past information we want to consider in our model or how many past observations we are willing to use for predicting the next value in the sequence. Two different values for this hyperparameter were considered in this study. The 10-fold Cross Validation method was used to compare the models more robustly and reliably. Different hyperparameter configurations were used for LSTM and Dense blocks. These configurations are shown in Table. 5.2. 60 models were trained in total. To simplify referring to each configuration, each model is coded based on its configuration as follow:

Model Label = LSTM and Dense Configurations (1-5 based on Table. 5.2)
+ Encoding method (No Encoder or 4-8-1 or 16-32-1)
+ Discrete or Continuous Event Detector (Dis or Con)
+ n_past value (5 or 10)

For example, if a model is labeled as $1 + 4 - 8 - 1 +$ Con $+$ n_past $= 5$, it means that the first configuration for LSTM and Dense Blocks based on Table. 5.2, the first encoder or $4 - 8 - 1$, continuous event detector, and n_past$= 5$ are used in this model.

We used C-index as the evaluation metric as discussed in the last chapter. The comparison between the models was based on the 10-fold cross-validated average and STD of the C-index. We compared the models in four different ways as follow:

(1) Discrete vs. Continuous Event Detector

Table 5.2: LSTM and Dense Blocks configurations

Configuration Label	# LSTM layers	# LSTM Neurons	LSTM Activation Functions	LSTM Dropout-LSTM Recurrent Dropout Rate	# Dense layers	# Dense Neurons	Dense Activation Function	Dense Dropout Rate
1	3	4-4-4	Tanh-Tanh-ReLU	0.2-0.2	-	-	-	-
2	2	16-16	ReLU	0.2-0.2	-	-	-	-
3	3	4-4-4	Tanh-Tanh-ReLU	0.2-0.2	3	16-32-16	ReLU	0.25
4	3	4-8-16	SeLU	0.2-0.2	3	16-32-16	ReLU	0.1
5	4	8-8-8-8	SeLU	0.2-0.2	-	-	-	-

(2) Encoder vs. No-Encoder

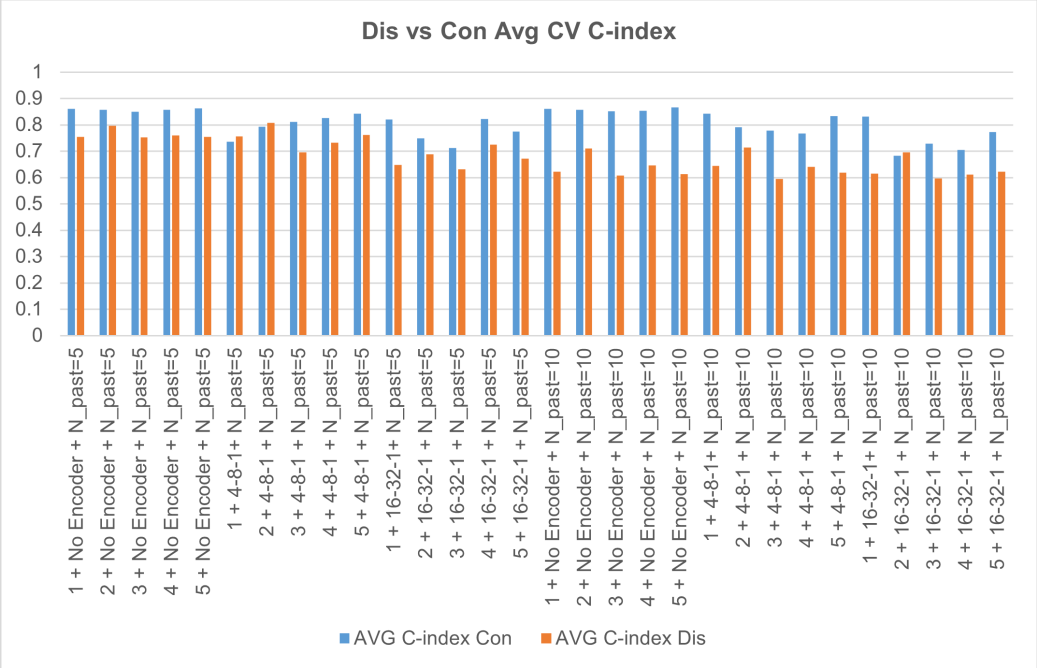
(3) n_past = 5 vs. n_past = 10

(4) Overall Comparison between all models

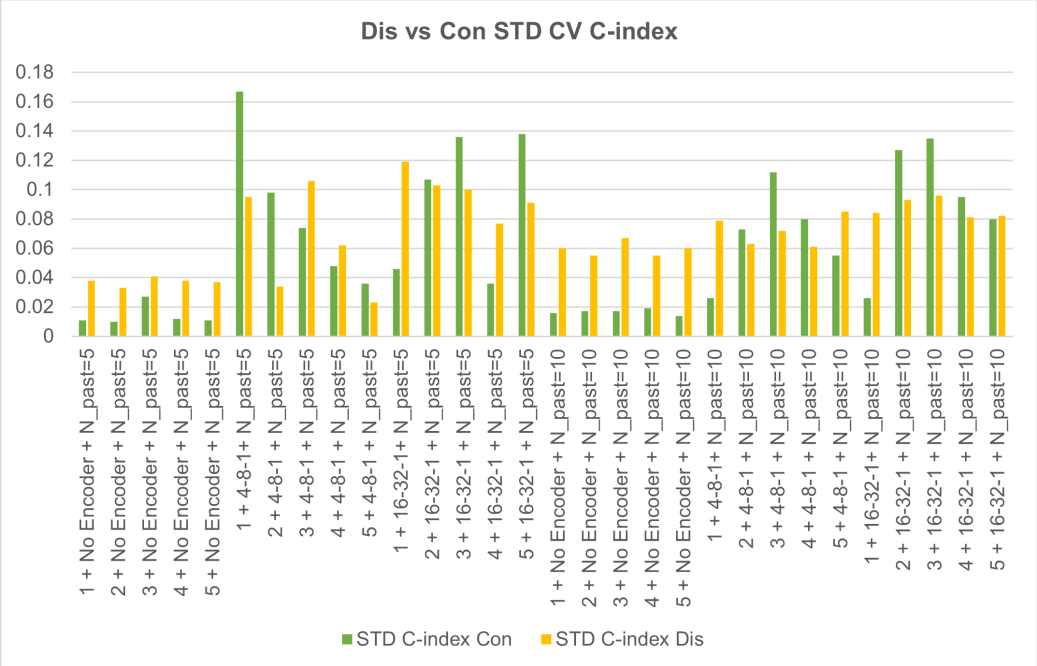
The performance of different models using discrete and continuous event detectors are illustrated in Fig. 5.6. The following observations could be concluded:

- Continuous event detector has a better average of cross-validated C-index in all configurations compared with the discrete event detector.
- Continuous event detector has a better STD of cross-validated C-index in 18 out of 30 configurations in comparison with the discrete event detector. It is also noteworthy to mention that we got the best STD result using a continuous event detector.

Thus, it is clear that using a continuous event detector outperformed the discrete event detector. That could be due to the fact that the input data used in this chapter were images. As we discussed earlier, there are no explicit and directly-measured features included in the model. Although each pixel is referred to as a feature by flattening the images, they are not like other values measured by sensors, like temperature, vibration, and force that might be more directly related to the system degradation. For that reason, more data might be needed in this case to extract such information. Using a continuous event detector provides this opportunity for us. Another comparison could be made between using and not using a dimension reduction technique, like Convolutional Autoencoder. The result of using and not using a Convolutional Autoencoder is shown in Fig. 5.7. The following results could be inferred from this figure:



(a) Average of cross-validated C-index result



(b) STD of cross-validated C-index result

Figure 5.6: Discrete versus continuous event detector performance comparison

- In most cases, using no encoder resulted in a better average of the cross-validated C-index. In a few cases, like $1 + Dis + n\_past = 5$ or $5 +$ Dis $+$ n_past $= 10$, using $4 - 8 - 1$ encoder resulted in a slightly better result.

- In the STD of the cross-validated C-index, using no encoders reflected better results in 19 out of 20 configurations. $4 - 8 - 1$ in comparison with $16 - 32 - 1$ encoder performed better, and in some cases, its STD was near using no encoder.

In conclusion, as the results suggest, if we use a well-tunned autoencoder, it has the ability to perform as well as using no encoder. We got the best overall result using no encoder. That is expectable since a dimension reduction technique is used. Although we are preserving part of important information by using a dimension reduction technique, we are still losing some information in exchange for reducing the input data which may affect the result as we have seen here. Having said that, the resulted C-index still offer that using a dimension reduction is acceptable, and might be helpful in other cases when we are dealing with a huge amount of high-dimensional data.

As we stated earlier, another influential hyperparameter in models containing LSTM layers is n_past. Two values for this hyperparameter were considered, 5 and 10, and a comparison between their results is shown in Fig. 5.8. Based on this figure, we can infer the following observations:

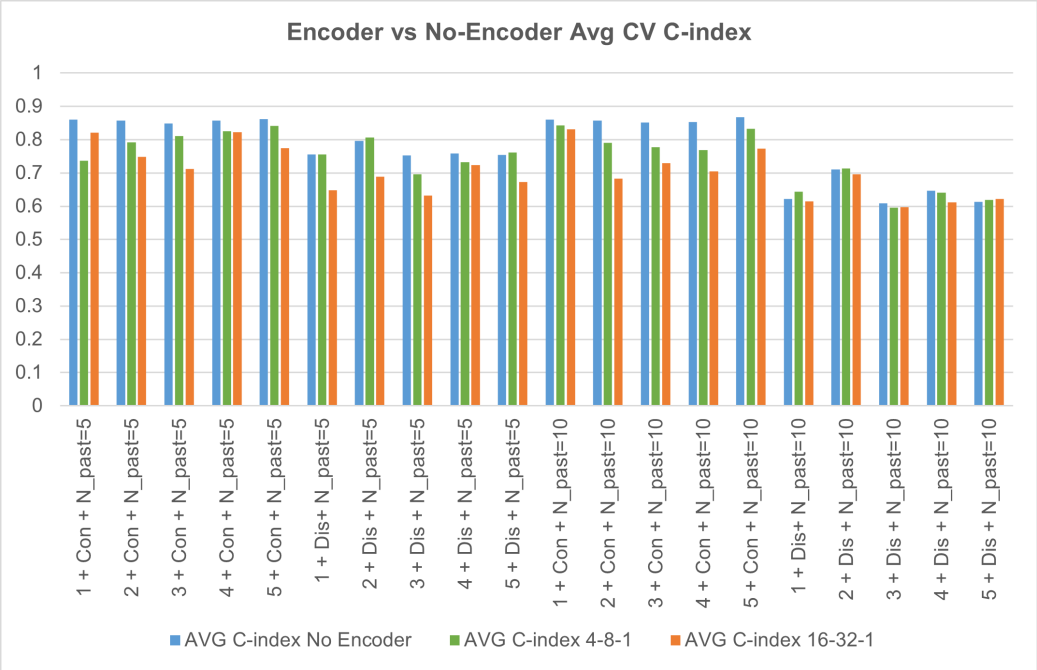
- Although the best model was achieved using both n_past$= 5$ and n_past$= 10$, in 24 out of 30 cases, n_past$= 5$ performed better or equal compared with n_past$= 10$.

- In 17 out of 30 cases, n_past$= 5$ had lower STD in comparison with n_past$= 10$. However, except in a few configurations, the STDs values were closed to each other.

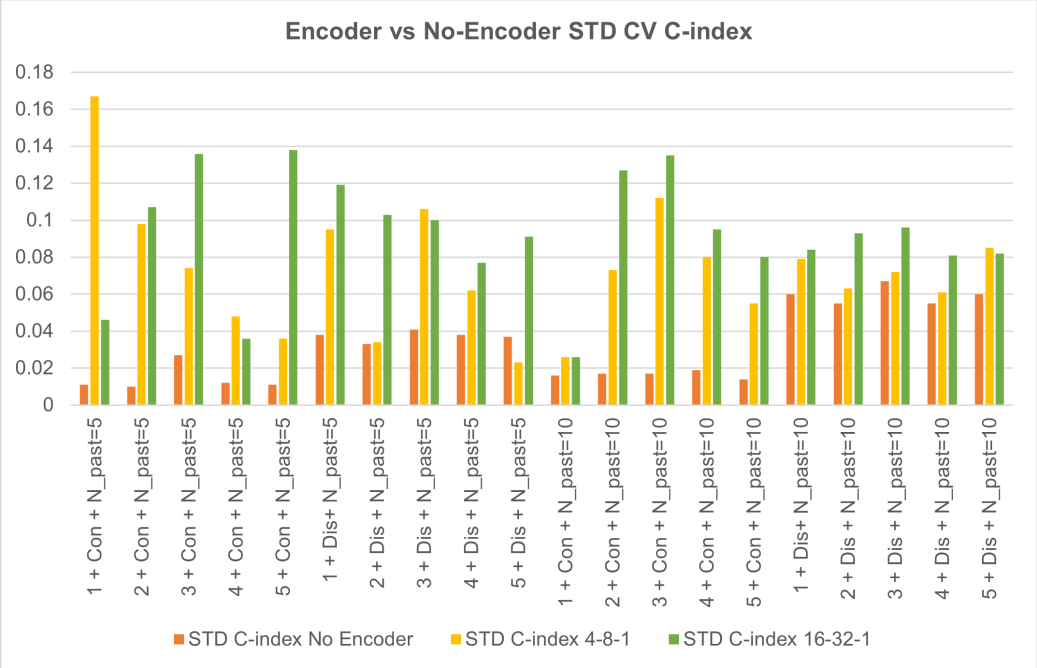
Therefore, $n\_past = 5$ might be a better value for this hyperparameter. Nevertheless, one can decide to try other values for this hyperparameter and get even better results.

Finally, an overall comparison between all the model configurations is shown in Fig. 5.9. The best top 5 configurations considering both the average and STD of the cross-validated C-index are as follow:

- $1 +$ No Encoder $+$ Con $+$ n_past $= 5$

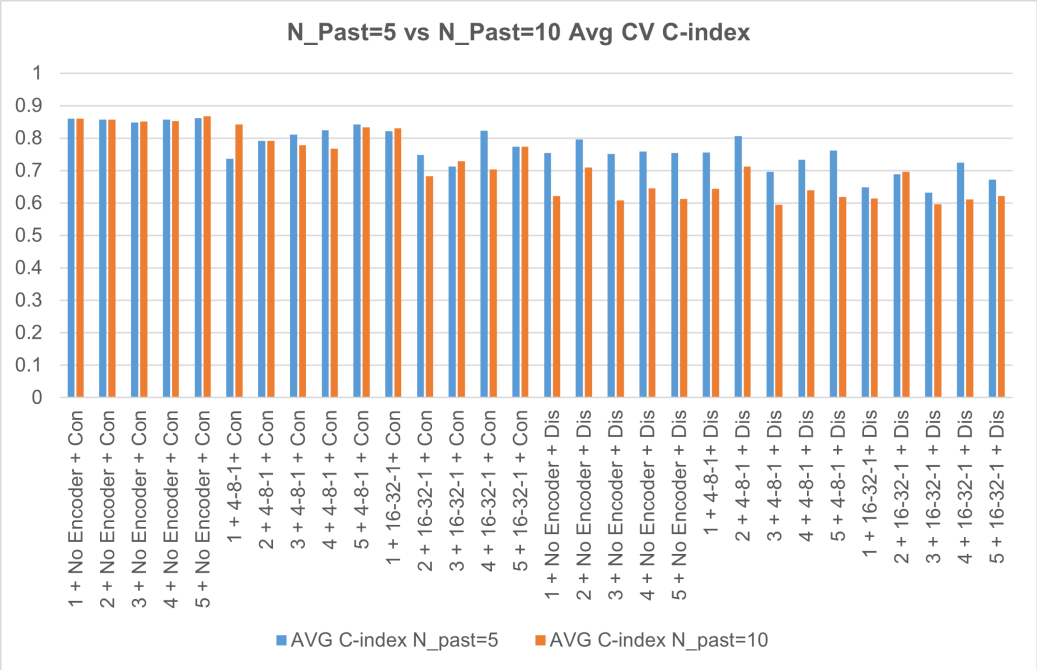


(a) Average of cross-validated C-index result

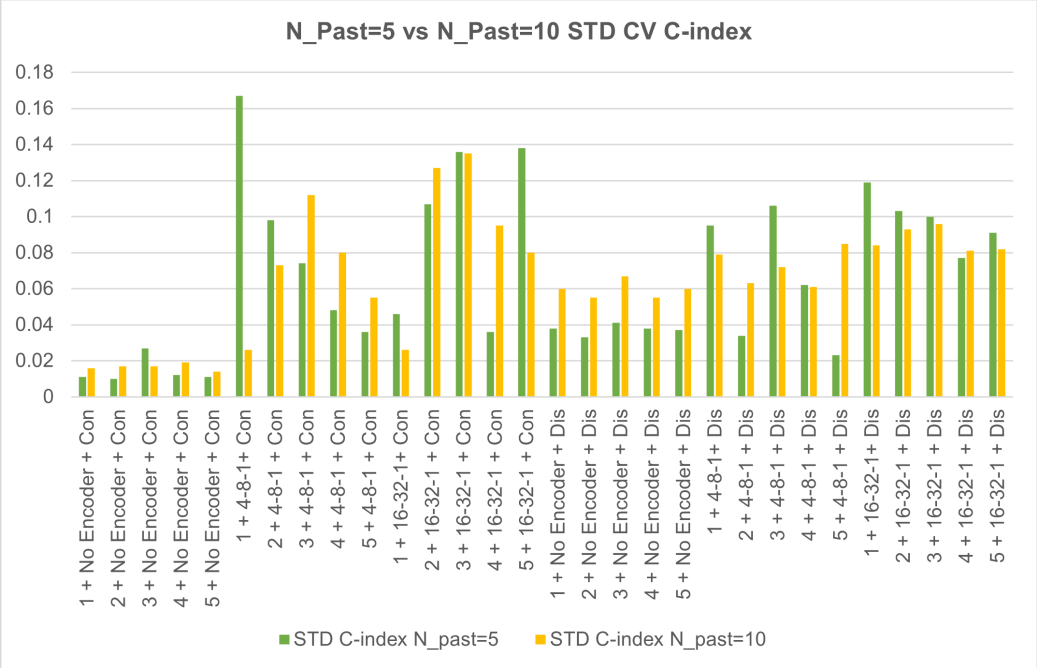


(b) STD of cross-validated C-index result

Figure 5.7: Encoder versus No Encoder performance comparison



(a) Average of cross-validated C-index result



(b) STD of cross-validated C-index result

Figure 5.8: n_past= 5 versus n_past= 10 performance comparison

- $2 +$ No Encoder $+$ Con $+$ n_past $= 5$
- $4 +$ No Encoder $+$ Con $+$ n_past $= 5$
- $5 +$ No Encoder $+$ Con $+$ n_past $= 5$
- $5 +$ No Encoder $+$ Con $+$ n_past $= 10$

For illustration purposes, nine different image streams were chosen from the test set, and the predicted hazard risk of these streams using the best model above, $1+$No Encoder$+$Con$+$n_past $=$ 5, is shown in Fig. 5.10. As it is clear in the figure, the predicted hazard risk is increasing over time which was expected. Another interesting point to mention in this figure is that those streams that failed earlier have greater hazard risk at a certain point of time, like $t = 10$, in comparison with the others that failed later.

5.4 Summary

Motivated by the emergence of high-dimensional data like images in the maintenance domain, this chapter proposed a novel method to predict the hazard risk of a system subjected to failure from the data captured by the imaging devices. The proposed model uses LSTM and Dense layers to enable the model to learn from the sequential data and time series nature of the data. To overcome the lack of failure data in the maintenance area that prevents us to train a DL-based model properly, a solution based on using a continuous event detector regarding the proximity of the observation to the system failure is provided that allows the model to learn from the censored data as well. Since working with low-dimensional data is easier compared with high-dimensional images, a dimension reduction technique based on the Convolutional Autoencoder is provided. Different models are trained for hyperparameter tuning purposes, and comparisons between these models are made based on the average and the standard deviation of the cross-validated C-index. The evaluation results illustrate the great potential of the proposed method.

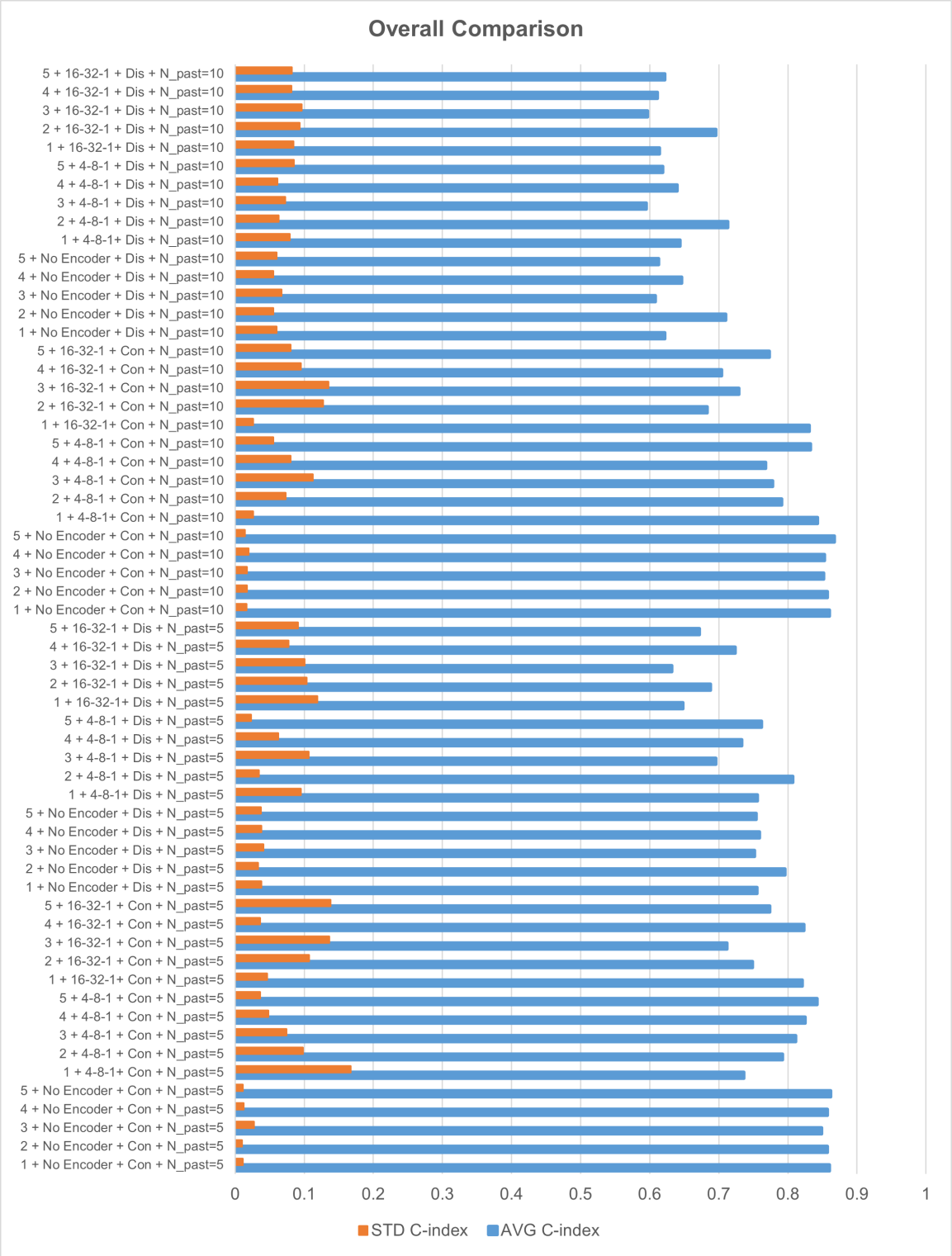


Figure 5.9: Performance comparison between all models

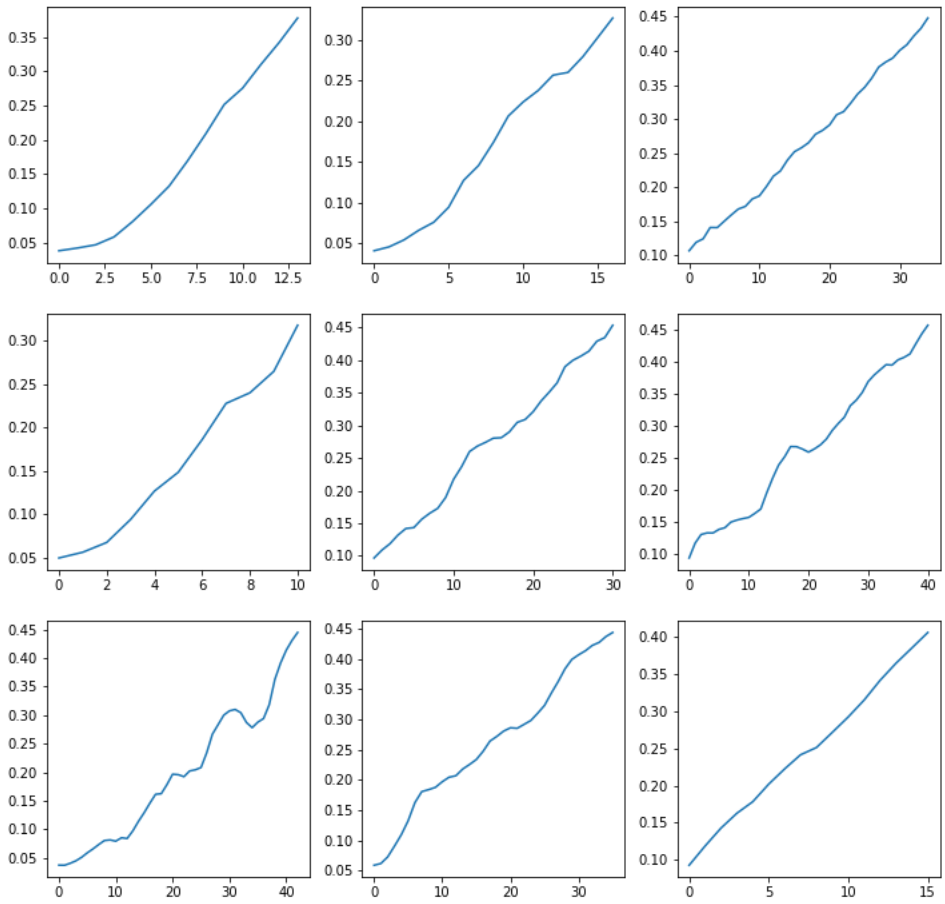


Figure 5.10: Predicted hazard risk for different image streams

Chapter 6

Summary and Future Research Directions

This chapter concludes the thesis and summarizes its contributions. We will also provide some insight for future works in this line of research at the end of this chapter.

6.1 Summary of Thesis Contributions

In this thesis, we proposed three case studies that could be used to develop a framework toward CBM for prognostic and diagnosis purposes. In summary, the objective of this study was to predict the hazard risk of a system subject to degradation and classify the health state of the system using both its age and some sensors' measurements. DL models as the state-of-the-art ML models were utilized in each case study.

(1) **Semi-Supervised Clustering-based Method for Fault Diagnosis and Prognosis: A Case Study:** We proposed a semi-supervised and hybrid framework to develop an intelligent MDSS in this study that augmented statistical, ML, and RL methods. Regarding the limitation of the RUL for maintenance decision-making, hazard rate was chosen as the primary decision variable. A time-varying extension of the classic Cox's PHM was used to estimate the hazard risk. Weibull distribution was used to model and estimate the baseline hazard. Next, a modified version of K-means clustering was used to extract the hidden state of the

system based on the calculated hazard risk and baseline hazard. Eventually, classification methods were utilized to predict the system's state for future use. Different classification methods were trained, and their decision graphs showed interesting results and a great potential to use the proposed methodology to develop an MDSS. Therefore, the RL method was employed to obtain an optimal maintenance policy based on the proposed methodology.

(2) **HazNet: Non Linear and Deep Hazard Architecture for Maintenance Management:** The second study proposed a novel non-linear extension of the Cox's PHM model to predict the system's hazard risk. Researchers have been using Cox's PHM to predict the hazard risk of a system because of its simplicity yet general usability. Nevertheless, Cox's PHM have some usage limitation. For example, a linear relationship between covariates is assumed in this model that might be a superficial assumption in many real-world problems. Moreover, Cox's PHM does not allow time-varying covariates. Thus, researchers proposed many approaches to extend the traditional Cox's PHM to allow a non-linear relationship between time-independent and time-varying covariates. In this regard, we proposed a non-linear DL-based variant of Cox's PHM for the first time in the maintenance context. To take time into account, we utilized three different kinds of DL models, namely LSTM, One-dimensional CNN, and a hybrid model combining both LSTM and One-dimensional CNN, that could be used for time-dependent datasets. DL models are powerful and state-of-the-art methods that could automate feature generation and learn complex features without human intervention and domain knowledge about the features. Moreover, as opposed to statistical methods, it has a remarkable ability to process massive amounts of data. Another method was proposed in this case study that empowers models to learn from the censored data. Previously, the researcher considered the event variable as a discrete variable. This variable shows whether a failure has happened or not (0 if failure has not happened or 1 if failure has happened). Since its value for all the censored data was equal to 0, the previous non-linear extensions to the Cox's PHM that take time-varying covariates into account could not learn from the censored data. Considering the lack of failure data in many cases, we are both affecting the model's training, and we are losing information by not using the censored data. Although failure data

contain much more enriched information about the degradation and failure of the system, censored data, especially the censored data right before the system failure, still have some valuable information that we can use to train a more accurate model. In this respect, we considered this variable a continuous variable and set its value for each observation based on how much it is close to the system's failure. Therefore, those censored data closer to the system's failure would contribute more to the model training than those far from the system failure. The proposed method was applied to the NASA Turbofan engine failure dataset. LSTM and hybrid models showed promising results. The results also suggested that using a continuous event detector yields reliable results when the model structure is more complex.

(3) **Deep Learning-based Survival Analysis for a System Subject to Stochastic Degradation using High-Dimensional Condition Monitoring Data:** One of the significant advantages of using DL over statistical and classical ML methods is that it can also analyze unstructured data like images. A framework to utilize the same method proposed in the first case study for images is developed in the second case study. The proposed method was applied to a simulated dataset, including images containing information about the system's degradation. Images are high-dimensional data that require heavier computations and more complex models to analyze. This problem was addressed using a DL-based dimension reduction technique known as Convolutional Autoencoder. Convolutional Autoencoder maps high-dimensional data into a lower-dimensional space that requires less memory and computation, and it is easier to work with. Additionally, we encountered the same issue as the first case study because we had fewer failure data in this dataset. The idea of using a continuous event detector was applied here too. Although the first case study results proposed that whenever the model gets complex, the continuous event detector tended to perform more robustly and reliably, the difference between discrete and continuous event detectors was not noticeable. However, the difference in the second case study was significant. That is because the models used to analyze images are more complex than the model used for standard data, and features should be learned from the images themselves as opposed to the dataset analyzed in the first case study that features were provided directly.

6.2 Future Research

Some of the potential future research directions are listed as follow:

(1) One possible future research could be to study continuous event detectors in different settings. As we have seen, introducing redundant data for model training can be a drawback of using such a method. One idea of improving this issue might be using a threshold in which we only contribute those censored data closer to the system's failure and not participate in those censored data in the earlier stage of the system run. Another fruitful research in this context is to generalize the idea of using continuous event detectors when no failure time is available. To compute the continuous event detectors in this setting, one might attempt to predict the failure time and then use those predicted failure times.

(2) Another research direction could also be using other maintenance policies in the MDSS development. For example, a maintenance policy with two sampling intervals proposed in [Naderkhani et al., 2017] can be studied.

(3) Last but not least, in real-world problems, structured and unstructured datasets might be available. Another interesting future work could be the study of the fusion of datasets with different types, like combining structured datasets with image data.

References

Accorsi, R., Manzini, R., Pascarella, P., Patella, M., and Sassi, S. (2017). Data mining and machine learning for condition-based maintenance. *Procedia manufacturing*, 11:1153–1161.

Alaswad, S. and Xiang, Y. (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability engineering & system safety*, 157:54–63.

Álvarez, C., López-Campos, M., Stegmaier, R., Mancilla-David, F., Schurch, R., and Angulo, A. (2019). A condition-based maintenance model including resource constraints on the number of inspections. *IEEE Transactions on Reliability*, 69(3):1165–1176.

Amruthnath, N. and Gupta, T. (2018). A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 355–361. IEEE.

Andriotis, C. and Papakonstantinou, K. (2021). Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliability Engineering & System Safety*, 212:107551.

Arab, A., Tekin, E., Khodaei, A., Khator, S. K., and Han, Z. (2016). System hardening and condition-based maintenance for electric power infrastructure under hurricane effects. *IEEE Transactions on Reliability*, 65(3):1457–1470.

Augustin, T. (2004). An exact corrected log-likelihood function for cox's proportional hazards model under measurement error and some extensions. *Scandinavian Journal of Statistics*, 31(1):43–50.

Aydemir, G. and Paynabar, K. (2019). Image-based prognostics using deep learning approach. *IEEE Transactions on Industrial Informatics*, 16(9):5956–5964.

Ayo-Imoru, R. and Cilliers, A. (2018). Continuous machine learning for abnormality identification to aid condition-based maintenance in nuclear power plant. *Annals of Nuclear Energy*, 118:61–70.

Babu, G. S., Zhao, P., and Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications*, pages 214–228. Springer.

Bai, Y., Sun, Z., Deng, J., Li, L., Long, J., and Li, C. (2018). Manufacturing quality prediction using intelligent learning approaches: A comparative study. *Sustainability*, 10(1):85.

Banjevic, D., Jardine, A., Makis, V., and Ennis, M. (2001). A control-limit policy and software for condition-based maintenance optimization. *INFOR: Information Systems and Operational Research*, 39(1):32–50.

Chen, C., Liu, Y., Wang, S., Sun, X., Di Cairano-Gilfedder, C., Titmus, S., and Syntetos, A. A. (2020). Predictive maintenance using cox proportional hazard deep learning. *Advanced Engineering Informatics*, 44:101054.

Chen, N., Chen, Y., Li, Z., Zhou, S., and Sievenpiper, C. (2011). Optimal variability sensitive condition-based maintenance with a cox ph model. *International journal of production research*, 49(7):2083–2100.

Coraddu, A., Oneto, L., Ghio, A., Savio, S., Anguita, D., and Figari, M. (2016). Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):136–153.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

Cox, D. R. (1975). Partial likelihood. *Biometrika*, 62(2):269–276.

Davidson-Pilon, C. (2021). lifelines, survival analysis in python.

Deutsch, J. and He, D. (2016). Using deep learning based approaches for bearing remaining useful

life prediction. In *Annual Conference of the PHM Society*, volume 8.

Deutsch, J. and He, D. (2017). Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):11–20.

Dozat, T. (2016). Incorporating nesterov momentum into adam.

Duan, C., Makis, V., and Deng, C. (2020). A two-level bayesian early fault detection for mechanical equipment subject to dependent failure modes. *Reliability Engineering & System Safety*, 193:106676.

Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., and Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183:240–251.

Fang, X., Paynabar, K., and Gebraeel, N. (2019). Image-based prognostics using penalized tensor regression. *Technometrics*, 61(3):369–384.

Faraggi, D. and Simon, R. (1995). A neural network model for survival data. *Statistics in medicine*, 14(1):73–82.

Harrell, F. E., Califf, R. M., Pryor, D. B., Lee, K. L., and Rosati, R. A. (1982). Evaluating the yield of medical tests. *Jama*, 247(18):2543–2546.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, J., Sun, Q., and Ye, Z.-S. (2020). Condition-based maintenance planning for systems subject to dependent soft and hard failures. *IEEE Transactions on Reliability*.

Huang, J., Chang, Q., and Arinez, J. (2020). Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, 160:113701.

Huang, J. and Wellner, J. A. (1997). Interval censored survival data: a review of recent progress. In *Proceedings of the First Seattle Symposium in Biostatistics*, pages 123–169. Springer.

Huynh, K. T. (2019). A hybrid condition-based maintenance model for deteriorating systems subject to nonmemoryless imperfect repairs and perfect replacements. *IEEE Transactions on*

Reliability, 69(2):781–815.

Jafari, L., Naderkhani, F., and Makis, V. (2017). Joint optimization of maintenance policy and inspection interval for a multi-unit series system using proportional hazards model. *Journal of the operational research society*, pages 1–12.

Jardine, A., Anderson, P., and Mann, D. (1987). Application of the weibull proportional hazards model to aircraft and marine engine failure data. *Quality and reliability engineering international*, 3(2):77–82.

Jardine, A., Makis, V., Banjevic, D., Braticevic, D., and Ennis, M. (1998). A decision optimization model for condition-based maintenance. *Journal of Quality in Maintenance Engineering*.

Jardine, A. K., Lin, D., and Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510.

Jardine, A. K. and Tsang, A. H. (2005). *Maintenance, replacement, and reliability: theory and applications*. CRC press.

Jia, Y. and Jeong, J.-H. (2020). Deep learning for quantile regression under right censoring: Deepquantreg. *arXiv preprint arXiv:2007.07056*.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):1–12.

Kleinbaum, D. G. and Klein, M. (2012). Extension of the cox proportional hazards model for time-dependent variables. In *Survival analysis*, pages 241–288. Springer.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. (2018). Sliced-wasserstein autoencoder: An embarrassingly simple generative model. *arXiv preprint arXiv:1804.01947*.

Koul, H., Susarla, V., and Van Ryzin, J. (1981). Regression analysis with randomly right-censored data. *The Annals of statistics*, pages 1276–1288.

Lall, P., Deshpande, S., and Nguyen, L. (2016). Ann based rul assessment for copper-aluminum wirebonds subjected to harsh environments. In *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–10. IEEE.

Lee, D. and Pan, R. (2017). Predictive maintenance of complex system with multi-level reliability

structure. *International Journal of Production Research*, 55(16):4785–4801.

Li, X., Ding, Q., and Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11.

Li, X., Zhang, W., and Ding, Q. (2019). Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. *Reliability engineering & system safety*, 182:208–218.

Li, Z., Zhou, S., Choubey, S., and Sievenpiper, C. (2007). Failure event prediction using the cox proportional hazard model driven by frequent failure signatures. *IIE transactions*, 39(3):303–315.

Lindley, D. V. (1958). Fiducial distributions and bayes' theorem. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 102–107.

Liu, Y., Chen, Y., and Jiang, T. (2020). Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach. *European Journal of Operational Research*, 283(1):166–181.

Lu, Y., Sun, L., Zhang, X., Feng, F., Kang, J., and Fu, G. (2018). Condition based maintenance optimization for offshore wind turbine considering opportunities based on neural network approach. *Applied Ocean Research*, 74:69–79.

Ma, M., Chen, X., Wang, S., Liu, Y., and Li, W. (2016). Bearing degradation assessment based on weibull distribution and deep belief network. In *2016 International Symposium on Flexible Automation (ISFA)*, pages 382–385. IEEE.

Makis, V. and Jardine, A. (1991). Computation of optimal policies in replacement models. *IMA Journal of Management Mathematics*, 3(3):169–175.

Makis, V. and Jardine, A. K. (1992). Optimal replacement in the proportional hazards model. *INFOR: Information Systems and Operational Research*, 30(1):172–183.

Mathew, V., Toby, T., Singh, V., Rao, B. M., and Kumar, M. G. (2017). Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning. In *2017 IEEE International Conference on Circuits and Systems (ICCS)*, pages 306–311. IEEE.

Miao, H., Li, B., Sun, C., and Liu, J. (2019). Joint learning of degradation assessment and rul prediction for aeroengines via dual-task deep lstm networks. *IEEE Transactions on Industrial*

Informatics, 15(9):5023–5032.

Miao, Q. and Makis, V. (2007). Condition monitoring and classification of rotating machinery using wavelets and hidden markov models. *Mechanical systems and signal processing*, 21(2):840–855.

Naderkhani, F., Jafari, L., and Makis, V. (2017). Optimal cbm policy with two sampling intervals. *Journal of Quality in Maintenance Engineering*.

Naderkhani, F. and Makis, V. (2016). Economic design of multivariate bayesian control chart with two sampling intervals. *International Journal of Production Economics*, 174:29–42.

Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., and Loncarski, J. (2018). Machine learning approach for predictive maintenance in industry 4.0. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE.

Park, K., Choi, Y., Choi, W. J., Ryu, H.-Y., and Kim, H. (2020). Lstm-based battery remaining useful life prediction with multi-channel charging profiles. *Ieee Access*, 8:20786–20798.

Qin, J. and Shen, Y. (2010). Statistical methods for analyzing right-censored length-biased data under cox model. *Biometrics*, 66(2):382–392.

Remadna, I., Terrissa, S. L., Zemouri, R., Ayad, S., and Zerhouni, N. (2020). Leveraging the power of the combination of cnn and bi-directional lstm networks for aircraft engine rul estimation. In *2020 Prognostics and Health Management Conference (PHM-Besançon)*, pages 116–121. IEEE.

Ren, L., Dong, J., Wang, X., Meng, Z., Zhao, L., and Deen, M. J. (2020). A data-driven auto-cnn-lstm prediction model for lithium-ion battery remaining useful life. *IEEE Transactions on Industrial Informatics*, 17(5):3478–3487.

Rengasamy, D., Jafari, M., Rothwell, B., Chen, X., and Figueredo, G. P. (2020). Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors*, 20(3):723.

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Rocchetta, R., Bellani, L., Compare, M., Zio, E., and Patelli, E. (2019). A reinforcement learning

framework for optimal operation and maintenance of power grids. *Applied energy*, 241:291–301.

Roy, T. S. and Vijay, A. H. (2019). A robust anomaly finder based on autoencoders. *arXiv preprint arXiv:1903.02032*.

Saon, S., Hiyama, T., et al. (2010). Predicting remaining useful life of rotating machinery based artificial neural network. *Computers & Mathematics with Applications*, 60(4):1078–1087.

Saxena, A., Goebel, K., Simon, D., and Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE.

Shi, Z. and Chehade, A. (2021). A dual-lstm framework combining change point detection and remaining useful life prediction. *Reliability Engineering & System Safety*, 205:107257.

Si, X., Li, T., Zhang, Q., and Hu, X. (2018). An optimal condition-based replacement method for systems with observed degradation signals. *IEEE Transactions on Reliability*, 67(3):1281–1293.

Sun, Q., Ye, Z.-S., and Chen, N. (2017). Optimal inspection and replacement policies for multi-unit systems subject to degradation. *IEEE Transactions on Reliability*, 67(1):401–413.

Tang, D., Makis, V., Jafari, L., and Yu, J. (2015). Optimal maintenance policy and residual life estimation for a slowly degrading system subject to condition monitoring. *Reliability Engineering & System Safety*, 134:198–207.

Tian, L., Zucker, D., and Wei, L. (2005). On the cox model with time-varying regression coefficients. *Journal of the American statistical Association*, 100(469):172–183.

Tian, Z., Jin, T., Wu, B., and Ding, F. (2011). Condition based maintenance optimization for wind power generation systems under continuous monitoring. *Renewable Energy*, 36(5):1502–1509.

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B., and Wei, L.-J. (2011). On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in medicine*, 30(10):1105–1117.

Verbert, K., De Schutter, B., and Babuška, R. (2018). A multiple-model reliability prediction approach for condition-based maintenance. *IEEE Transactions on Reliability*, 67(3):1364–1376.

Wang, J., Wen, G., Yang, S., and Liu, Y. (2018). Remaining useful life estimation in prognostics using deep bidirectional lstm neural network. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 1037–1042. IEEE.

Wang, M., Cheng, J., and Zhai, H. (2020). Life prediction for machinery components based on cnn-bilstm network and attention model. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 851–855. IEEE.

Wang, Q., Zheng, S., Farahat, A., Serita, S., and Gupta, C. (2019). Remaining useful life estimation using functional data analysis. In *2019 ieee international conference on prognostics and health management (icphm)*, pages 1–8. IEEE.

Wei, S., Bao, Y., and Li, H. (2020). Optimal policy for structure maintenance: A deep reinforcement learning framework. *Structural Safety*, 83:101906.

Wong, E. L., Jefferis, T., and Montgomery, N. (2010). Proportional hazards modeling of engine failures in military vehicles. *Journal of Quality in Maintenance Engineering*.

Wong, G. Y., Osborne, M. P., Diao, Q., and Yu, Q. (2017). Piecewise cox models with right-censored data. *Communications in Statistics-Simulation and Computation*, 46(10):7894–7908.

Wu, B., Tian, Z., and Chen, M. (2013). Condition-based maintenance optimization using neural network-based health condition prediction. *Quality and Reliability Engineering International*, 29(8):1151–1163.

Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495.

Wu, X. and Ryan, S. M. (2010). Value of condition monitoring for optimal replacement in the proportional hazards model with continuous degradation. *IIE Transactions*, 42(8):553–563.

Wu, Y., Lin, Y., Lu, S.-E., Li, C.-S., and Shih, W. J. (2014). Extension of a cox proportional hazards cure model when cure information is partially known. *Biostatistics*, 15(3):540–554.

Xia, F., Ning, J., and Huang, X. (2018). Empirical comparison of the breslow estimator and the kalbfleisch prentice estimator for survival functions. *Journal of biometrics & biostatistics*, 9(2).

Xu, J. and Duraisamy, K. (2020). Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and*

Engineering, 372:113379.

Xu, Z. and Saleh, J. H. (2021). Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, page 107530.

Yang, B., Liu, R., and Zio, E. (2019). Remaining useful life prediction based on a double-convolutional neural network architecture. *IEEE Transactions on Industrial Electronics*, 66(12):9521–9530.

Yang, H., Li, W., and Wang, B. (2021). Joint optimization of preventive maintenance and production scheduling for multi-state production systems based on reinforcement learning. *Reliability Engineering & System Safety*, 214:107713.

Yang, L., Ma, X., and Zhao, Y. (2017). A condition-based maintenance model for a three-state system subject to degradation and environmental shocks. *Computers & Industrial Engineering*, 105:210–222.

Yao, L., Dong, Q., Jiang, J., and Ni, F. (2020). Deep reinforcement learning for long-term pavement maintenance planning. *Computer-Aided Civil and Infrastructure Engineering*, 35(11):1230–1245.

Yu, W., Kim, I. Y., and Mechefske, C. (2020). An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering & System Safety*, 199:106926.

Zhan, Y., Makis, V., and Jardine, A. K. (2006). Adaptive state detection of gearboxes under varying load conditions based on parametric modelling. *Mechanical Systems and Signal Processing*, 20(1):188–221.

Zhang, C., Lim, P., Qin, A. K., and Tan, K. C. (2016). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10):2306–2318.

Zhang, N. and Si, W. (2020). Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. *Reliability Engineering & System Safety*, 203:107094.

Zhao, G., Zhang, G., Ge, Q., and Liu, X. (2016). Research advances in fault diagnosis and prognostic based on deep learning. In *2016 Prognostics and system health management conference (PHM-Chengdu)*, pages 1–6. IEEE.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237.

Zhao, S., Makis, V., Chen, S., and Li, Y. (2018). Health assessment method for electronic components subject to condition monitoring and hard failure. *IEEE Transactions on Instrumentation and Measurement*, 68(1):138–150.

Zheng, R., Chen, B., and Gu, L. (2020). Condition-based maintenance with dynamic thresholds for a system using the proportional hazards model. *Reliability Engineering & System Safety*, 204:107123.

Zheng, R. and Makis, V. (2020). Optimal condition-based maintenance with general repair and two dependent failure modes. *Computers & Industrial Engineering*, 141:106322.

Zheng, S., Ristovski, K., Farahat, A., and Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *2017 IEEE international conference on prognostics and health management (ICPHM)*, pages 88–95. IEEE.

Zhou, P. and Yin, P. (2019). An opportunistic condition-based maintenance strategy for offshore wind farm based on predictive analytics. *Renewable and Sustainable Energy Reviews*, 109:1–9.

Zhu, J., Chen, N., and Shen, C. (2019). A new deep transfer learning method for bearing fault diagnosis under different working conditions. *IEEE Sensors Journal*, 20(15):8394–8402.